THE ANALYSIS OF FLOW IN PIPE NETWORK SYSTEMS

(vol 2)

by

PAUL ERIC PREECE

APPENDICES

To a thesis presented at the

University of Aston in Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

October 1972

Table of Contents

APPENDIX (1)

THE USE OF A, B, C, D AND F MATRICES IN COMPUTER PROGRAMS

The storage of network information and the speed at which it may be handled in a computer program is of great importance, especially in the analysis of large scale problems. The five network describing matrices are sparse and contain only +1s, -1s and 0s, and special techniques may be implemented which go some way towards minimising the amount of data to be stored and handled, and maximising the speed at which the information can be used.

The proposed method of storing the $\underline{A}$ matrix is a modification of that used by Branin (21) and other workers. It relies on referencing of the branches of the network as discussed in Chapter (3). The $\underline{A}$ matrix need not be stored as a complete array. All the necessary information about the $\underline{A}$ matrix may be stored as a two dimensional list. The list is made up of an array which has the dimensions number of branches by two. The rows of the array correspond to the referenced branches of the network and the columns to the two terminal nodes of any branch. Each row contains one positive and one negative integer, corresponding respectively to the reference number of the node towards which the branch is directed, and the reference number of the node away from which the branch is directed. The advantage of using this method is that only a small amount of information has to be handled. Most other users of the list technique use a further column which contains the reference number of the branch. If however the input data about the branches is arranged is ascending order of reference, this requirement is obviated and a two column array only is needed. From a computational point it is also better to reference the link branches with the highest numbers for the purpose of multiplying other matrices more easily by the submatrices of $\underline{A}$. It is also advantageous to include in the list the restriction that the reference node

with the highest absolute value is placed in the right hand column. This is not a necessity, but it gives rise to a reduction in computing time used when performing multiplications. An example of the input data list for $\underline{A}$ of the network of figure (3.2.1) is shown in figure (A.1).

The $\underline{B}$ matrix may be made available to a program in either of two ways. It may be input in its fullest form as data by the user. This is a long and complicated procedure since a path has to be traced out from every node to the datum node, at the same time noting the direction of every branch in the node to datum path. The procedure is open to error. The information has to be recorded on data preparation sheets and then transferred to card or tape input. Alternatively the $\underline{B}$ matrix in an already partitioned form may be set up within the program, using the information contained in the branch-node connection list. This may be achieved using the relationship between $\underline{A}_T$ and $\underline{B}_T$ which has previously been described.

It has been indicated that the $\underline{A}_T$ matrix can, if necessary, be constructed from the branch node connection list. However, the $\underline{A}_T$ matrix is not symmetric nor has it any other features, apart from its sparsity, which will enable its inversion to be speeded up. An alternative method of automatically generating the $\underline{B}_T$ matrix was investigated and compared with the computational requirements of the standard inversion of $\underline{A}_T$. The method is not susceptible to any errors other than those already present in the connection list. It also acts as a verifier of input data, since if any of the meshes of the network are not closed loops, the user is informed by a statement output from the computer and computation ceases. The subroutine which implements the method is called SEARCHBT and is included in the program listings in appendix (12).

Basically the method involves the setting up of a matrix which contains information about the way in which the branches are connected. This information is successively searched until all the node to datum

Figure (A.1)

The branch node connection list of the graph shown in figure (3.2.1)

1	-2
2	-3
-2	5
-5	7
6	-7
7	-8
4	-5
5	-9
-1	4
-4	6
-3	9
8	-9

paths have been found. The matrix NCP (Nodal Connection Panel) of dimensions number of nodes by number of nodes is set up from the information in the branch-node connection list. The elements of the matrix NCP are set up such that NCP (i,j) is a positive or negative integer value corresponding to the reference number of the branch which runs between nodes i and j. For example, if branch 4 runs between nodes 2 and 6, then NCP (2,6) = -NCP (6,2) = 4.

The completed matrix is antisymmetric about a zero diagonal, and each column or row contains a list of the branch reference numbers, with the direction indicated, which are connected to the node corresponding to the row or column. $\underline{B}_T$ is made up of tree branches only and therefore information about link branches can be ommitted from NCP.

A further matrix used in SEARCHBT is the PR matrix (Path Record). This is used to keep a record of the reference numbers of the branches in any particular node to datum path.

Using the complete NCP matrix a search is started at the head of the column which corresponds to the datum node (as mentioned previously, conveniently numbered the highest). The column is searched until a non-zero entry is encountered. The value of the non-zero entry together with its sign is stored in the first element of PR. The search is then re-started at the head of the column corresponding to the row in which the last non-zero entry was encountered. Each time the program encounters a non-zero entry its value is compared with the preceding non-zero entry. If the sum of the two values is zero, then a terminal node has been reached and the path record stored in PR is transferred to a column of $\underline{B}_T$. The entries in NCP corresponding to the terminal nodes are then deleted and the search recommenced at the head of the datum node column. The process is repeated until all non-zero entries are deleted or until sufficient node to datum paths are found. The implication of this is that the program may be employed to produce a variety of results.

If NCP is set up from a partitioned connection list using tree branches only, then the $\underline{B}_T$ matrix produced by the subroutine will be exactly that corresponding to the matrix which would have been constructed by the user as a result of the partitioning. Alternatively, if NCP is set up from a non partitioned connection list, the $\underline{B}_T$ matrix produced will be identical to the $\underline{B}_T$ matrix set up for the condition of maximum overlap.

Once evaluated, the $\underline{B}_T$ matrix need not be stored as a full matrix. It can be stored as a series of lists, each one corresponding to one node to datum path containing only non zero entries.

Using a Honeywell 316, a 22 node network was examined and the following times for the evaluation of the $\underline{B}_T$ matrix from a connection list were obtained:

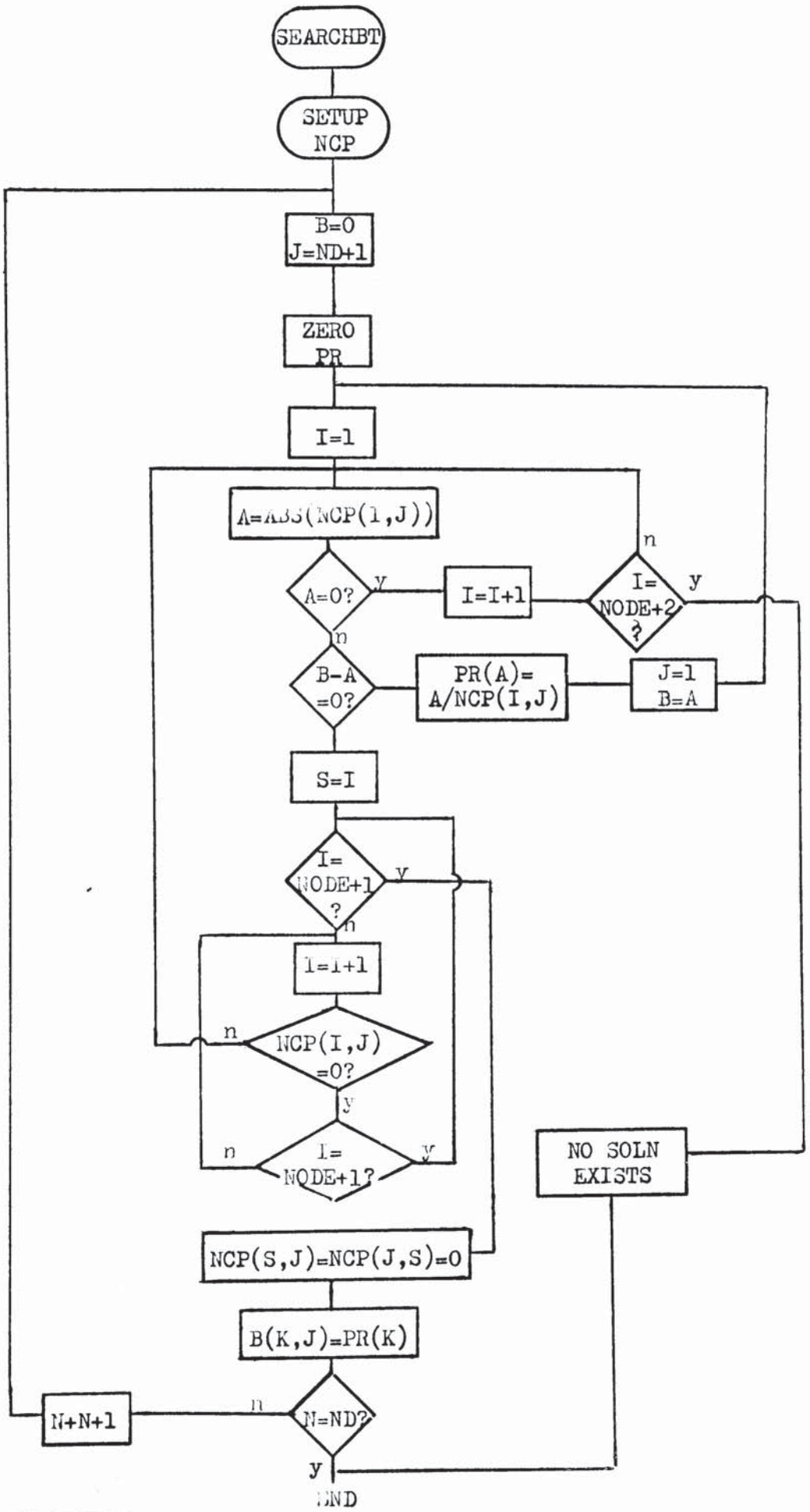
SEARCHBT	45 seconds
Standard Inversion	90 seconds

The flowchart which describes the operation of SEARCHBT is shown in figure (A.2).

The preparation of the $\underline{C}$ matrix by a user is also a tedious process which is open to error. A method of generating the $\underline{C}$ matrix within a program from a simple connection list is proposed. The link partitioned part $\underline{C}_L$ is easily generated since it is a unit matrix of dimension number of links by number of links. It may only be recognised in this form if the link branches have the highest reference numbers. The tree partitioned part may be generated using equation (3.3.6). As discussed, $\underline{A}_L$ need not be stored as a full matrix and $\underline{B}_T$ can be generated from the tree partition of the branch-node connection list. $\underline{C}$ itself need not be stored as a full matrix. It can be stored as two lists, one within the other. Every row of the list contains in the first column, the number of branches in the corresponding mesh and the following elements in the row are a list of branch reference numbers, with directions indicated, that

Figure (A.2)

The flowsheet for SEARCHBT



make up the mesh. This method of storage is used for the two Hardy Cross programs. The value of retaining $\underline{A}_L$ in list form is indicated by analysing the total number of matrix multiplications required to produce the $\underline{C}_T$ matrix.

For the graph of figure (3.2.1):

by full matrix multiplication	256 operations
by using $\underline{A}_L$ in list form	64 operations

and for the TESTI network:

by full matrix multiplication	7497 operations
by using $\underline{A}_L$ in list form	714 operations

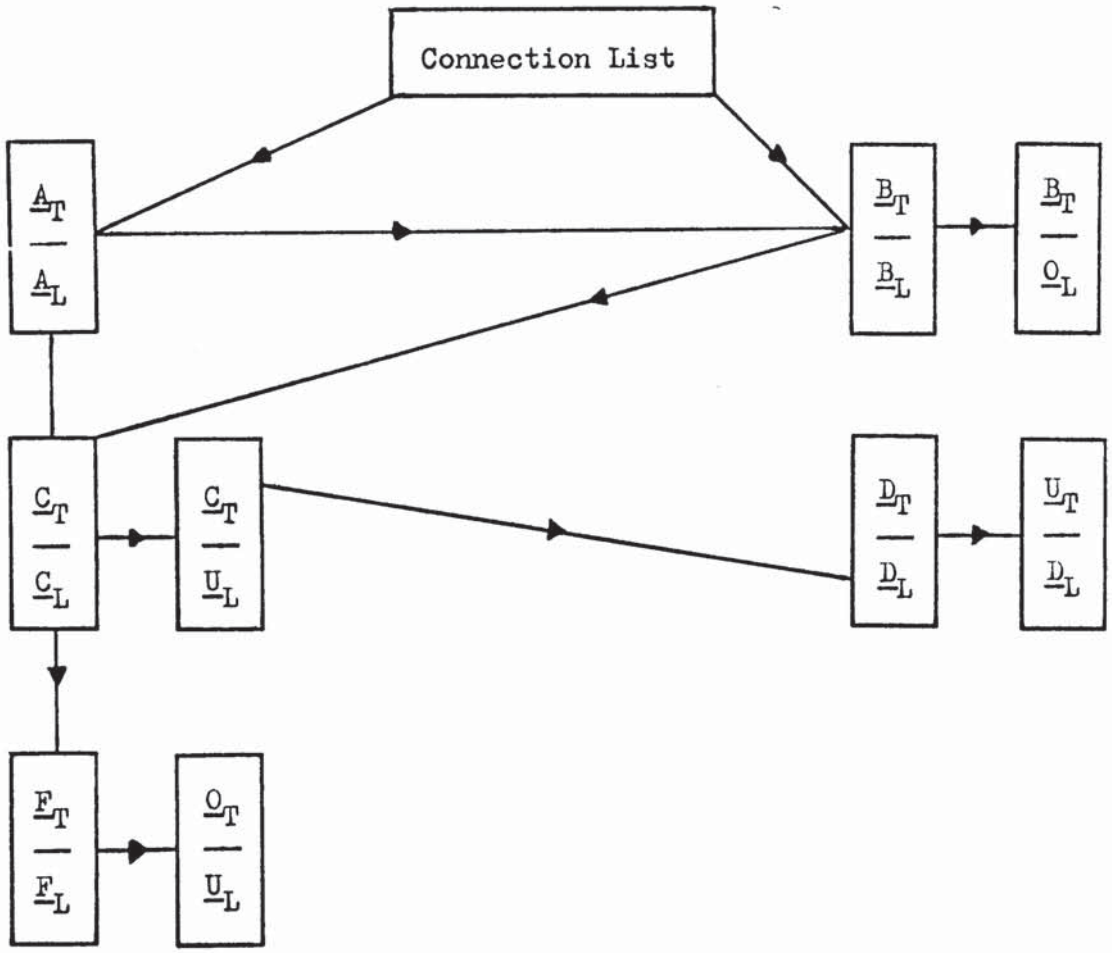
Special techniques may also be used for the establishment of the $\underline{D}$ matrix. The tree partition part $\underline{D}_T$ is a unit matrix and therefore need never be set up. The $\underline{D}_L$ partitioned part is easily obtained from $\underline{C}_T$ according to equation (3.3.8).

The $\underline{F}$ matrix, being composed of a zero submatrix $\underline{F}_T$ and a unit matrix $\underline{F}_L$, need never be stored or set up as a full matrix within a machine.

Figure (A.3) shows a flow chart which summarises the above discussion on the setting up of the network describing matrices from a simple connection list, and shows how every matrix may be constructed by following the steps indicated by the arrows.

Figure (A.3)

Flow chart for setting up any network describing matrix from a connection list



APPENDIX (2)

THE USE OF THE D MATRIX IN NETWORK ANALYSIS

A modification of the classical nodal method developed by Branin (19) but which has received little attention is the cutset method. Using the $\underline{D}$ matrix as an alternative to the $\underline{A}$ matrix an equivalent solution equation for the tree potentials is developed. The equation is given by

$$\underline{e}_T' = (\underline{\check{D}}_T.\underline{Y}_T.\underline{D}_T + \underline{\check{D}}_L.\underline{Y}_L.\underline{D}_L)^{-1}\ \underline{\check{D}}.(\underline{I} - \underline{Y}.\underline{E})$$

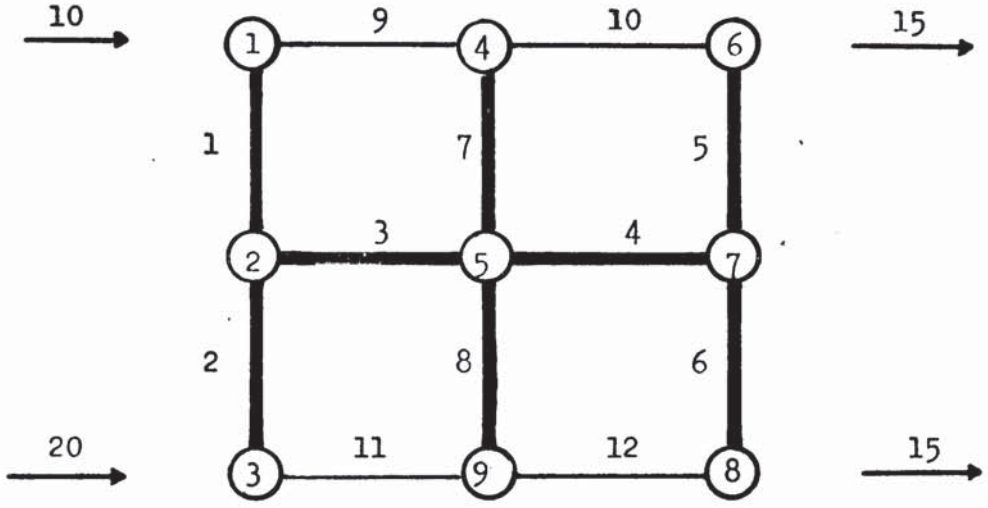
No published computer program which implements this equation is available for comparison. It is felt by the author that there is no advantage to be gained by the use of the equation, since the size of the solution matrix to be inverted is the same as that of the standard node to datum method using the $\underline{A}$ matrix. However it is arguable that a small advantage may be gained from the fact that $\underline{D}_T = \underline{U}_T$ and therefore computation of the $(\underline{\check{D}Y\check{D}})$ matrix may be accomplished more readily than the $(\underline{\check{A}Y\check{A}})$ matrix. However it has been shown in appendix (1) that the $\underline{A}$ matrix need not be stored in its complete form and that any multiplication by $\underline{A}$ may be achieved by simple successive additions and subtractions.

For the above reason no computer program which utilises the above solution equation has been written by the author. However any analysis of network solution methods must be comprehensive and the inclusion of the above equation as the basis of a solution method shows that there are a variety of solution methods available, of which some are more useful than others.

APPENDIX (3)

THE SOLUTION OF A NETWORK PROBLEM USING THE ORTHOGONAL MESH SOLUTION METHOD

The implementation of the orthogonal mesh solution method relies on the solution of equation (3.8.12) which gives the individual mesh flows of a network, from which the individual branch flows and hence the individual branch pressure drops and node to datum pressure drops may be calculated. Consider the following network with the nodal demand and supply vector specified as shown:



The $\underline{B}$ and $\underline{C}$ matrices are given in Chapter (3) for this network.

The solution of equation (3.8.12) necessitates the formation of the $(\underline{E}'_L - \check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}')$ matrix. There are no pumps in the system and therefore $\underline{E}'_L = \underline{0}$. $(\check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}')$ may be formed from $(\check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{I}_T)$ where $\underline{I}_T$ is given by

$$
\begin{matrix}
10 \\
-20 \\
-30 \\
-30 \\
-15 \\
15 \\
0 \\
0
\end{matrix}
$$

and $\underline{Z}_T$ is given by

4

5

6

6

5

8

5

7

The negative of $(\breve{\underline{C}}_T.\underline{Z}_T.I_T)$ is therefore given by

-220

-255

-280

-300

The mesh solution matrix may be formed by simple matrix multiplications and the addition of $\underline{Z}_L$ where $\underline{Z}_L$ is given by

4

4

6

7

to give $(\breve{\underline{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)$

19.0	-5.0	6.0	0.0
-5.0	20.0	0.0	6.0
6.0	0.0	24.0	-7.0
0.0	6.0	-7.0	28.0

The inverse of the mesh solution equation $(\breve{\underline{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)^{-1}$ is given by

0.0632	0.0184	-0.0813	-0.0085
0.0184	0.0591	-0.0089	-0.0149
-0.0813	-0.0089	0.0505	0.0145
-0.0085	-0.0149	0.0145	0.0425

The individual mesh flows are given by

$$(\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L)^{-1} \cdot - (\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{I}_T)$$

which gives for $\underline{i}'$

-10.92

-12.157

-12.195

-11.141

The flows in the tree branches due to the mesh flows may be calculated from $\underline{C}_T\ \underline{i}'$ to give for $\underline{i}_T$

-10.92

-12.19

23.17

23.29

12.15

-11.14

-1.23

-1.05

Addition of the flows in the tree branches due to the node to datum flows and the flows $\underline{i}_T$ produce the total individual tree branch flows. From a knowledge of these flows the nodal potentials $\underline{e}'$ may be calculated to give

30.75

34.44

73.47

-12.94

-7.01

-62.28

-47.83

-78.32

These node to datum potentials can be seen to compare exactly (within the limits of calculation) with the results obtained in appendices (7) and (5).

APPENDIX (4)

A COMPARISON BETWEEN THE CLASSICAL NODAL METHOD AND THE NODAL METHOD OF ORTHOGONAL NETWORKS

It has been shown in Chapter (3) that the primitive pump terms may be treated as equivalent current sources using equation (3.4.6) in a modified form as shown below

$$\underline{Y}.\underline{e} = \underline{i} + (\underline{I} - \underline{Y}.\underline{E})$$

where $(\underline{I} - \underline{Y}.\underline{E})$ is treated as a new total current source. With this modification the orthogonal equations in $\underline{\alpha}$ then become

$$\begin{matrix} \underline{I}' \\ \hline \underline{i}' \end{matrix} = \underline{\breve{\alpha}}.\underline{Y}.\underline{\alpha} \cdot \begin{matrix} \underline{e}' \\ \hline \underline{0} \end{matrix}$$

where

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

the equivalent current sources in the primitive network having been transformed to node to datum orthogonal quantities. The modified orthogonal equation in $\underline{\alpha}$ when solved in terms of $\underline{e}'$ gives

$$\underline{e}' = (\underline{\breve{A}}_T.\underline{Y}_T.\underline{A}_T + \underline{\breve{A}}_L.\underline{Y}_L.\underline{A}_L)^{-1}.\underline{I}'$$

which when expanded for comparison with the classical nodal method becomes

$$\underline{e}' = (\underline{\breve{A}}_T.\underline{Y}_T.\underline{A}_T + \underline{\breve{A}}_L.\underline{Y}_L.\underline{A}_L)^{-1}.\underline{\breve{A}}(\underline{I} - \underline{YE})$$

where

$$(\underline{\breve{A}}_T.\underline{Y}_T.\underline{A}_T + \underline{\breve{A}}_L.\underline{Y}_L.\underline{A}_L) = (\underline{\breve{A}}\underline{Y}\underline{A})$$

The equation in $\underline{e}'$ developed out of the orthogonal approach can be seen to be identical to that shown by Middleton (6) for the classical method due to Roth (13).

APPENDIX (5)

THE SOLUTION OF A NETWORK PROBLEM USING THE ORTHOGONAL NODAL METHOD

The implementation of the orthogonal nodal solution method relies on the solution of a modified form of equation (3.8.15). The primitive voltage sources are assumed to have been transformed to equivalent current sources. The complete solution equations to equation (3.8.15) give equations in $\underline{i}'$ and $\underline{e}'$, only the equation in $\underline{e}'$ need be considered for orthogonal nodal studies. This equation in $\underline{e}'$ has been compared to the classical nodal equation in appendix (4).

The network problem to be solved is identical to that given in appendix (3). The branch node connection list may be formed by reference to the directions of the numbered branches incident at the numbered nodes. It has been indicated that to set up the matrix $(\underline{\check{A}}_T \cdot \underline{Y}_T \cdot \underline{A}_T + \underline{\check{A}}_L \cdot \underline{Y}_L \cdot \underline{A}_L)$ the full matrix multiplications need not be performed. From the information in the branch-node connection list and the admittance vector $\underline{Y}$ the nodal solution matrix is given by:

0.5	-0.25	0	-0.25	0	0	0	0
-0.25	0.616	-0.2	0	-0.166	0	0	0
0	-0.2	0.366	0	0	0	0	0
-0.25	0	0	0.7	-0.2	-0.25	0	0
0	-0.166	0	-0.2	0.676	0	-0.166	0
0	0	0	-0.25	0	0.45	-0.2	0
0	0	0	0	-0.166	-0.2	0.492	-0.125
0	0	0	0	0	0	-0.125	0.2679

The inverse of the nodal solution matrix $(\underline{\check{A}}_T \cdot \underline{Y}_T \cdot \underline{A}_T + \underline{\check{A}}_L \cdot \underline{Y}_L \cdot \underline{A}_L)^{-1}$ is therefore given by:

6.27	4.09	2.23	4.45	3.04	3.77	2.91	1.36
4.09	4.89	2.67	3.28	2.77	2.89	2.40	1.12
2.23	2.67	4.18	1.79	1.51	1.58	1.31	0.61
4.45	3.28	1.79	5.63	3.32	4.64	3.42	1.60
3.04	2.77	1.51	3.32	3.87	3.15	2.94	1.37
3.77	2.89	1.58	4.64	3.15	6.72	4.31	2.01
2.91	2.40	1.31	3.42	2.94	4.31	5.43	2.58
1.36	1.12	0.61	1.60	1.37	2.01	2.58	4.91

Since there are no pumps in the system the matrix $\breve{A}.(\underline{I} - \underline{Y}.\underline{E})$ may be represented in the very simple form of the node to datum flow vector $\underline{I}'$. This matrix premultiplied by the inverse of the nodal solution matrix yields a vector of node to datum potentials $\underline{e}'$ which is given by:

30.45
34.14
73.18
-13.24
-7.19
-61.77
-47.43
-78.12

The vector $\underline{e}'$ premultiplied by $\underline{A}$ gives the overall branch pressure drop $\underline{V}$ minus the effect of pump sources $\underline{E}$. Since there are no pumps $\underline{e}$ represents the overall branch pressure drops. These are

-3.69
-39.04
-41.33
-14.34
30.34

-6.05

-7.19

-43.69

-48.53

-73.18

-78.12

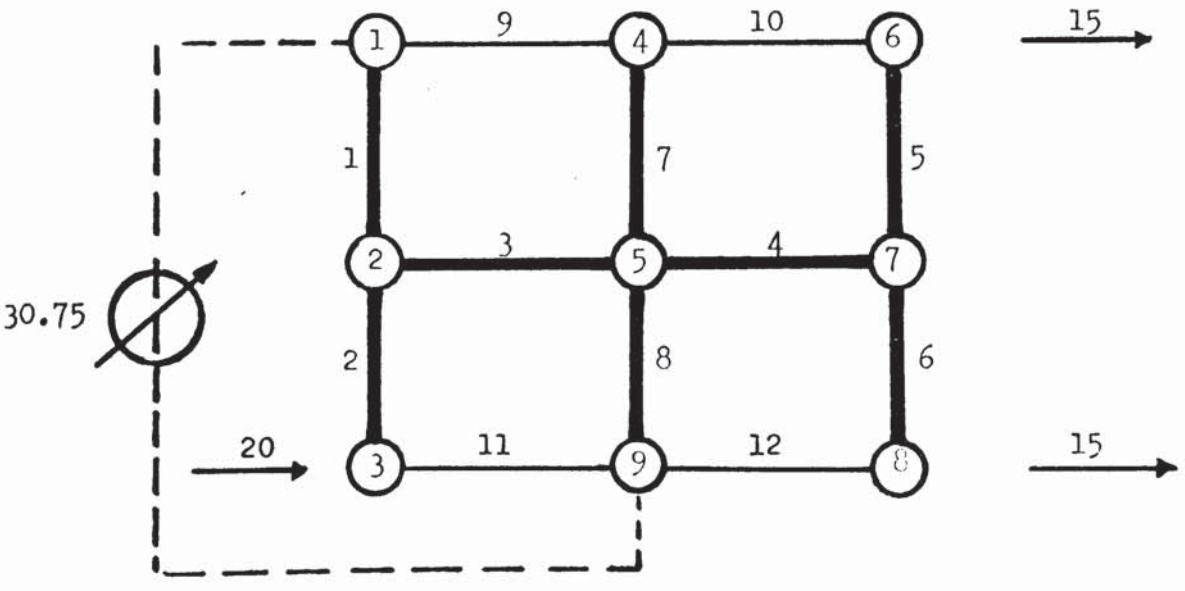
These can be seen to compare favourably with the branch pressure drops calculated for the same network system by other methods (cf the tree branch pressure drops of appendix (9)). The small differences in the equivalent values are probably attributable to the fact that the branch pressure drops of this appendix were calculated by a small computer program rather than by hand.

APPENDIX (6)

A WORKED EXAMPLE SHOWING THE SOLUTION OF A NETWORK PROBLEM WHICH HAS MORE THAN ONE OF ITS NODAL PRESSURES SPECIFIED

The individual branch flows and nodal pressures for the network shown below have been calculated by a variety of methods as shown in appendices (3) and (9). In all cases the nodal demand and supply vector has been fully specified. It is shown below that up to M (where M is the number of meshes) nodal pressures may be specified together with ND (where ND is the number of node to datum paths) inputs or outputs for any network, and a solution obtained in terms of the individual branch flows for that network.

From a knowledge of the individual nodal pressures calculated in appendix (3) the following problem may be set up. Assume that the supply of 10 units of flow at node (1) is unspecified but that a nodal pressure of 30.75 units is specified as an alternative, and that all other physical details of the network are identical with those of the network in appendix (3). The solution to this problem should be identical to the solutions obtained in appendices (3) and (9), since the value of 30.75 units for the nodal pressure of node (1) has been taken from those solutions.



The method proposed for the solution of this type of problem necessitates the inclusion of as many fictitious branches on the graph of the network as there are pressure specified nodes (excluding the datum node).

The fictitious branches are assumed to contain pressure sources which can bring about the required pressure at the pressure specified nodes. The fictitious branches are also assumed to interconnect the pressure specified nodes and the datum node as shown on the graph above. The direction associated with the fictitious branches is governed by the direction of pressure rise due to the proposed pressure source. (The direction of a branch is assumed to be in the opposite direction to the assumed pressure rise.) In the graph above node (1) is assumed to be at a pressure of 30.75 units greater than the datum and the direction of the fictitious branch is therefore towards the datum node. Each fictitious branch will be a link branch and therefore will create on the graph an additional mesh. Each fictitious branch is further assumed to have a zero impedance.

It is proposed that the network problem as outlined above may be solved by using equation (3.8.12) which is the equivalent of the classical mesh solution equation,

$$\underline{i}' = (\underline{\check{C}}_T.\underline{Z}_T.\underline{C}_T + \underline{Z}_L)^{-1}.(\underline{E}'_L - \underline{\check{C}}_T.\underline{Z}_T.\underline{B}_T.I') \qquad (3.8.12)$$

The nodal demand and supply vector $\underline{I}'$ is given by:

$$\underline{I}' = \begin{matrix} 0 \\ 0 \\ 20 \\ 0 \\ 0 \\ -15 \\ 0 \\ -15 \end{matrix}$$

and the node to datum vector $\underline{I}_T$ is given by $\underline{B}_T\underline{I}'$

$$\underline{B}_T \cdot \underline{I}' = \begin{matrix} 0 \\ -20 \\ -20 \\ -30 \\ -15 \\ 15 \\ 0 \\ -10 \end{matrix}$$

$\underline{Z}_T$ is given by

$$\begin{matrix} 4 \\ 5 \\ 6 \\ 6 \\ 5 \\ 8 \\ 5 \\ 7 \end{matrix}$$

and therefore $\underline{Z}_T\underline{I}_T$ is given by:

$$\begin{matrix} 0 \\ -100 \\ -120 \\ -180 \\ -75 \\ 120 \\ 0 \\ -70 \end{matrix}$$

The $\underline{C}$ matrix for the graph above may be constructed as shown below.

$$\underline{C} = \begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}$$

and therefore $\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{I}_T$ is given by:

$$\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{I}_T = \begin{array}{c} 120 \\ 255 \\ 150 \\ 370 \\ 50 \end{array}$$

$\underline{E}'_L$ is constructed by $\breve{\underline{C}} \cdot \underline{E}$ and is given by

$$\underline{E}'_L = \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 30.75 \end{array}$$

and therefore $(\underline{E}'_L - \breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}')$ is given by

$$(\underline{E}'_L - \breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{B}_T \cdot \underline{I}') = \begin{array}{c} -120 \\ -255 \\ -150 \\ -370 \\ -19.25 \end{array}$$

$(\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L)$ may be constructed by simple matrix multiplication and is given by:

$$(\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L) = \begin{bmatrix} 19 & -5 & 6 & 0 & 10 \\ -5 & 20 & 0 & 6 & 0 \\ 6 & 0 & 24 & -7 & 13 \\ 0 & 6 & -7 & 28 & -7 \\ 10 & 0 & 13 & -7 & 17 \end{bmatrix}$$

The inverse of $(\breve{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L)$ must be found, preferably using a computer, and is given by:

$$\begin{bmatrix} 0.0961 & 0.0307 & 0.0086 & -0.0225 & -0.0724 \\ 0.0307 & 0.0637 & 0.0012 & -0.0201 & -0.0273 \\ 0.0086 & 0.0012 & 0.0724 & 0.0030 & -0.0592 \\ 0.0225 & 0.0201 & 0.0030 & 0.0485 & 0.0309 \\ -0.0724 & -0.0273 & -0.0592 & 0.0309 & 0.1594 \end{bmatrix}$$

All the information required for the solution of equation (3.8.12) is available and the mesh flow vector $\underline{i}'$ is given by:

$$\begin{bmatrix} -10.94 \\ -12.13 \\ -12.19 \\ -11.14 \\ 10.05 \end{bmatrix}$$

The first four elements of this vector represent the individual mesh flows of the original meshes before the addition of the fictitious branch. They can be seen to compare exactly (within the limits of calculation) with the individual mesh flows calculated in appendix (3).

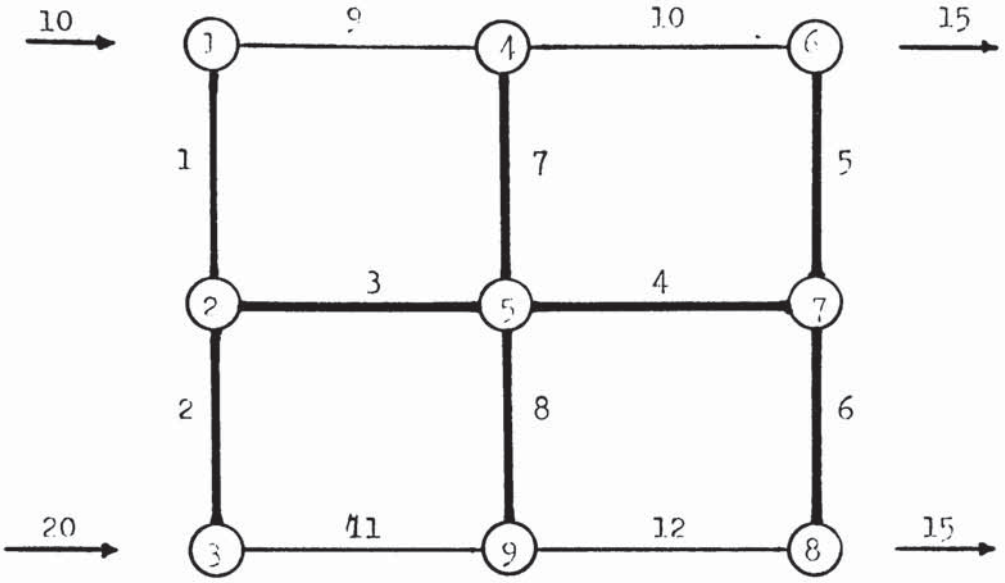
The fifth element of the vector $\underline{i}'$ represents the mesh flow through the mesh created by the fictitious branch. Its value of 10 units of flow entering the network at node (1) exactly balances the inputs and outputs of the system, and is equal to the unspecified nodal flow at node (1).

The same working may be extended to include more than one pressure

specified node, and further to the situation where both the nodal pressure and the nodal flow may be specified for any node.

APPENDIX (7)

A WORKED EXAMPLE OF NODAL DIAKOPTICS SHOWING THE PRINCIPLE OF TWO COMPONENT VOLTAGE ADDITION AT LOW-LEVEL



The low-level interconnection principle, which is extended to high-level interconnection, is equivalent to interconnecting all link branches of a network into the tree branches in one operation. No algorithm has been proposed for low-level interconnection but since the details of this appendix are similar to those of appendix (3), the reader can refer to the text to follow the steps.

The $\underline{B}$ and $\underline{C}$ matrices for the above network are given in appendix (3). The component of voltage $\underline{V}^{(1)}$ due to the external sources is given by $\underline{Z}_T \cdot \underline{I}_T$ where $\underline{I}_T$ is obtained from $\underline{B}_T \cdot \underline{I}'$ and is given by,

$$
\begin{matrix}
10 \\
-20 \\
-30 \\
-30 \\
-15 \\
15 \\
0 \\
0
\end{matrix}
$$

and $\underline{Z}_T$ is given by

4

5

6

6

5

8

5

7

and therefore $\underline{V}^{(1)}$ is given by

40

-100

-180

-180

-75

120

0

0

The second component $\underline{V}^{(2)}$ which is due to the interconnection of the link branches must be found. $\underline{V}_L^{(1)}$ is found from $\underline{\breve{C}}_T.\underline{V}^{(1)}$ to give

220

255

280

300

and since there are no pumps in the system, $\underline{E}_L^*$ which is given by $\underline{E}_L' - \underline{V}_L^{(1)}$ reduces to $-\underline{V}_L^{(1)}$. $\underline{V}^{(2)}$ can then be calculated from i, which is itself obtained from $\underline{i}'$ as shown below,

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

where $\underline{Z}_L = \begin{matrix} 4 \\ 4 \\ 6 \\ 7 \end{matrix}$ and $\underline{\breve{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T = \begin{matrix} 15 & -5 & 6 & 0 \\ -5 & 16 & 0 & 6 \\ 6 & 0 & 18 & -7 \\ 0 & 6 & -7 & 21 \end{matrix}$

therefore $(\underline{\breve{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L)^{-1} = \begin{matrix} 0.0632 & 0.0184 & -0.0183 & -0.0085 \\ 0.0184 & 0.0591 & -0.0089 & -0.0149 \\ -0.0183 & -0.0089 & 0.0505 & 0.0145 \\ -0.0085 & -0.0149 & 0.0145 & 0.0425 \end{matrix}$

therefore $\underline{i}' = \begin{matrix} -10.922 \\ -12.157 \\ -12.195 \\ -11.141 \end{matrix}$

i_T is given by $\underline{C}_T \cdot i'$ as shown below,

-10.92
-12.19
 23.17
 23.29
 12.15
-11.14
 -1.23
 -1.05

The component of flow $\underline{V}^{(2)}$ is given by $\underline{Z}_T \underline{i}_T$

-43.68
 60.97
138.70
139.78
 60.78
-89.12

-6.17

-7.38

The overall node to datum potential is obtained by only one further addition of the two voltage components $\underline{v}^{(1)}$ and $\underline{v}^{(2)}$ since there are no pumps in the system. Thus $\underline{e}$ is given by

-3.68

-39.03

-41.20

-15.22

-30.88

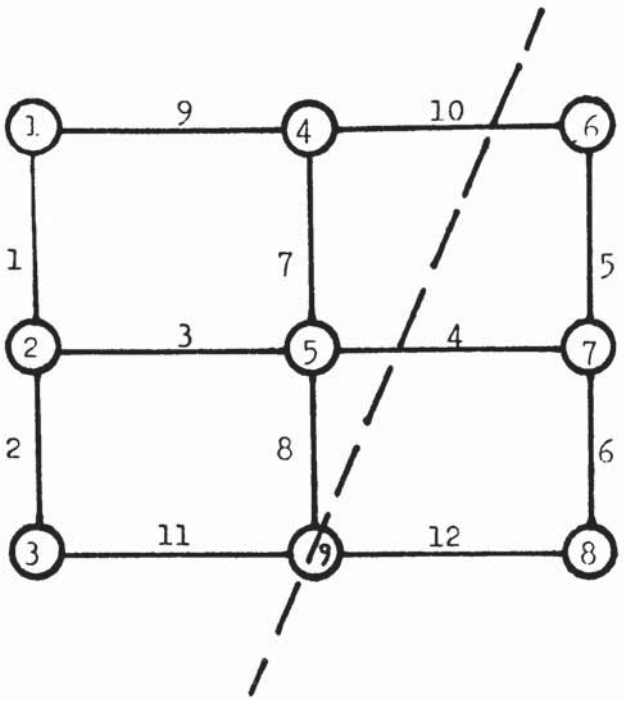
-6.17

-7.38

This represents the individual tree branch potential, premultiplication by $\underline{\breve{B}}_T$ would give the node to datum potential $\underline{e}'$ which is the same, within the limits of calculation, as that given in appendix (8). The dual voltage component concept has been shown to be a valid way of solving network problems and can be compared with the orthogonal mesh method. Notice though, that the number of multiplications and additions is less for the above method that for the orthogonal mesh method.

APPENDIX (8)

A WORKED EXAMPLE OF NODAL DIAKOPTICS SHOWING THE PRINCIPLE OF TWO COMPONENT VOLTAGE ADDITION OF HIGH LEVEL INTERCONNECTION

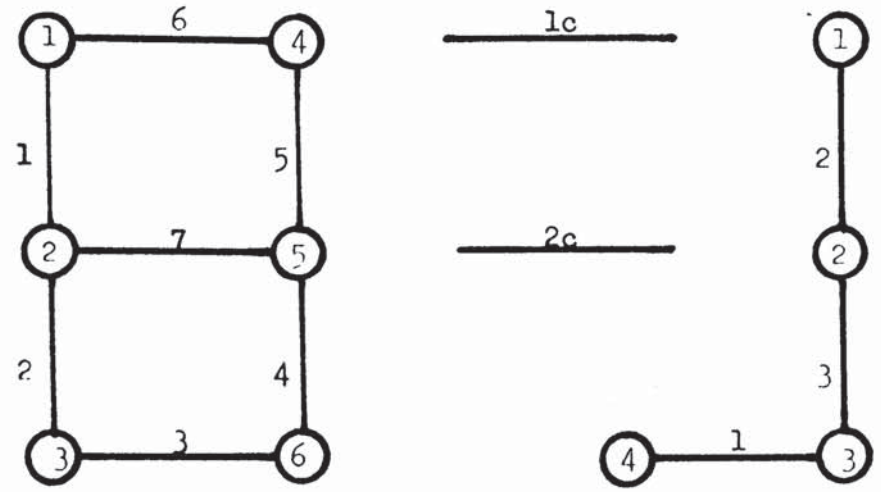


It is proposed to show the solution of the network problem outlined above by means of the nodal diakoptics method. The network is divided into two subnetworks by means of the cut represented by the dashed line. The network could be torn into a variety of subdivisions, but the cut proposed will serve to illuminate some important concepts. For instance if a further cut were made through branches 9 and 11, the network would be divided into three subnetworks and the reconnection would involve the inversion of a 4x4 matrix. The same inversion is required for solving the identical network problem by the orthogonal mesh method and therefore it would be difficult to see the advantages of the diakoptic method. With the cut shown above the network problem will be solved by the inversion of a 2x2 and a 5x5 matrix.

The algorithm shown in figure (3.10.1) is used as the basis of the solution method.

For reasons of clarity and computational simplicity it is necessary to reallocate the nodal and branch reference numbers of each subdivision and cut branch. Each branch and node of the network then has a local reference number and a global reference number. The subnetworks and cut

branches with their local reference numbers are shown below:



It is necessary to set up the new network information in terms of the subnetworks. The $\underline{A}$ matrix for each subnetwork is shown below. The smaller of the two subnetworks has no link branches.

$$\underline{A}_T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

$$\underline{A}_L = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{A}_T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

The impedance vector $\underline{Z}$ for each subnetwork and the two cut branches is given by:

4	7	4
5	5	6
6	8	
7		
5		
4		
6		

The nodal demand and supply vector for each subnetwork is given by:

10	-15
0	0
20	-15
0	
0	

With the available information about the network amassed, step 1 of the algorithm for nodal diakoptics may be completed. There are no pumps present in this network and therefore $\underline{I}^*$ for each subnetwork is given by

10	-15
0	0
20	-15
0	
0	

The second step requires the calculation of the nodal solution matrix $(\underline{\breve{A}}_T \cdot \underline{Y}_T \cdot \underline{A}_T + \underline{\breve{A}}_L \cdot \underline{Y}_L \cdot \underline{A}_L)$ for each subnetwork. For the larger subnetwork this may be carried out by the method advocated in appendix (5), which does not require the full matrix multiplications to be carried out. The inverse of the nodal solution matrix must then be calculated by any suitable inversion technique.

For the smaller subnetwork the inverse of the nodal solution matrix may be calculated directly. The relationship between $\underline{A}_T$ and $\underline{B}_T$ allows the following relationship to be set up:

$$(\underline{\breve{A}}_T \cdot \underline{Y}_T \cdot \underline{A}_T)^{-1} = (\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$$

Since the smaller subnetwork contains no link branches its inverse may be therefore readily calculated provided the $\underline{B}_T$ matrix is available. The $\underline{B}_T$ matrix for the smaller subnetwork is given by:

$$\underline{B}_T = \begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{matrix}$$

The inverses of the two solution matrices are given by:

$$\begin{matrix} 7.350 & 4.896 & 2.675 & 5.805 & 3.873 \\ 4.896 & 5.527 & 3.020 & 4.264 & 3.474 \\ 2.675 & 3.020 & 4.383 & 2.330 & 1.899 \\ 5.805 & 4.264 & 2.330 & 7.346 & 4.273 \\ 3.873 & 3.474 & 1.899 & 4.273 & 4.772 \end{matrix} \qquad \begin{matrix} 20.0 & 15.0 & 7.0 \\ 15.0 & 15.0 & 7.0 \\ 7.0 & 7.0 & 7.0 \end{matrix}$$

$\underline{V}^{(1)}$ for each subnetwork is then given by:

$$\begin{matrix} 127.00 \\ 109.35 \\ 114.40 \\ 104.65 \\ 76.71 \end{matrix} \qquad \begin{matrix} -405.0 \\ -330.0 \\ -210.0 \end{matrix}$$

Step 3 of the algorithm is omitted since no pump sources are present in the cut branches.

To transform the equivalent radial potentials $\underline{V}^{(1)}$ to connected orthogonal variables it is necessary to have available the transformation tensor $\underline{\gamma}^{**}_{TL}$ which shows the incidence of the cut branches at the locally referenced nodes. $\underline{\gamma}^{**}_{TL}$ for the network above is given by:

$$\underline{\gamma}^{**}_{TL} = \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{matrix}$$

$\underline{V}^{(1)}_L$ is then given by:

$$\underline{V}^{(1)}_L = \begin{matrix} -509.65 \\ -406.71 \end{matrix}$$

Step 5 of the algorithm which gives $\underline{E}^*$ is readily achieved since $\underline{E}'_L = 0$ and $\underline{E}^*$ is therefore negative of $\underline{V}_L^{(1)}$.

Step 6 of the algorithm, which interconnects the voltage effects due to the individual equivalent radial subnetworks, requires the computation of $(\underline{\breve{Y}}^{**}_{TL} \cdot (\underline{\breve{A}}.\underline{Y}.\underline{A})^{-1} . \underline{Y}^{**}_{TL}) + \underline{Z}_L)^{-1}$ where $\underline{Z}_L$ represents the impedance of the cut branches. In the computer program developed to utilise this method, the above multiplication is carried out in a sequential manner. $(\underline{\breve{A}}.\underline{Y}.\underline{A})$ for each subnetwork is multiplied by the appropriate submatrices $\underline{Y}^{**}_{TL}$ and $\underline{Y}^{**}_{TL}$ and the individual results summed. This means that information about one subnetwork only has to be handled at any one time in the main core store of a computer, thus reducing the amount of storage required.

The inverse of the overall solution matrix for the problem above is given by:

$$\begin{matrix} 0.0590 & -0.04416 \\ -0.04416 & 0.07812 \end{matrix}$$

$\underline{i}'$ of step 6 is then given by:

$$\begin{matrix} 12.109 \\ 6.703 \end{matrix}$$

$\underline{i}_T$ of step 7 is given by:

$$\begin{matrix} 0 \\ 0 \\ 0 \\ -12.109 \\ -6.703 \\ 12.109 \\ 6.703 \\ 0 \end{matrix}$$

and $\underline{V}^{(2)}$ the vector of node to datum potentials due to interconnection of step 8 is given by:

-96.25

-74.91

-40.93

-117.59

-83.72

342.72

282.17

131.68

The solution of the node to datum potentials $\underline{V}$ of step 9 is given by:

30.75

34.44

73.47

-12.94

-7.01

-62.28

-47.83

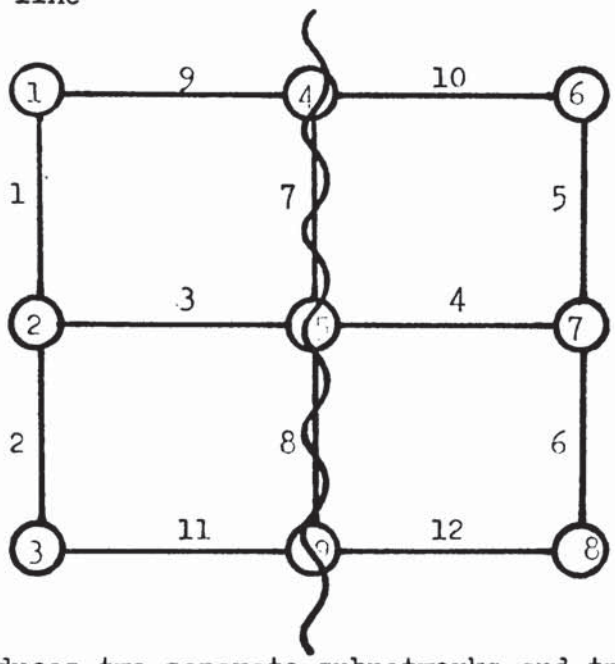
-78.32

From a knowledge of these node to datum potentials, premultiplication by $\underline{A}$ yields the individual branch pressure drops from which the individual branch flows $\underline{J}$ may be calculated. The node to datum potentials shown above produce exactly the same individual branch flows as those calculated by the other methods of network analysis shown in appendices (9) and (5).

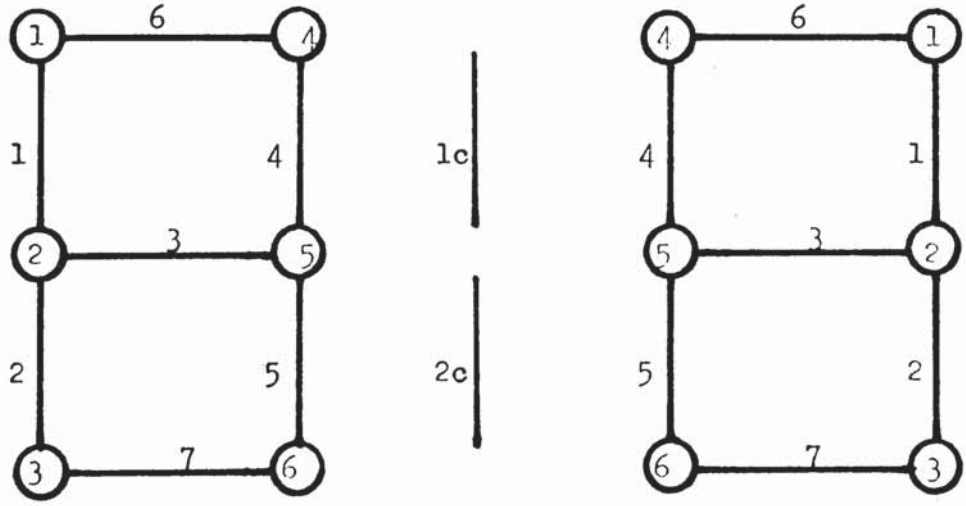
APPENDIX (9)

A WORKED EXAMPLE IN THE MESH METHOD OF DIAKOPTICS

For the solution to the network shown below by the mesh diakoptic method, it is proposed that the network will be torn apart along the axis of the curved line



This produces two separate subnetworks and two cut branches which with their local reference numbers are shown below.



The principle of the removal of the cut branches from a network to be solved by mesh diakoptics has been discussed in Chapter (3). In the above network the cut branches (7 and 8 of the original network) have been included in the cut sections to demonstrate the continuity of the individual meshes. However, these branches (4 and 5 in both subnetworks) must be thought of as short circuits having zero impedance. Only as cut

branches 1c and 2c do they have a positive impedance. The result is that the $\underline{C}$ matrix for each of the subnetworks may have a zero entry at the element position corresponding to the cut branch and the $\underline{Z}$ matrix for each of the subnetworks may also have a zero entry at the element position corresponding to the cut branch.

For the solution proposed below it is not necessary that both $\underline{C}$ and $\underline{Z}$ for each subnetwork have zero entries at the cut branch positions. A zero entry in either matrix will suffice.

The algorithm for the solution of network problems by mesh diakoptics given in figure (3.10.2) is followed as closely as possible in the working below.

The $\underline{C}_T$ matrix for each of the subdivisions is given by:

$$\begin{matrix} 1 & 0 \\ 0 & -1 \\ -1 & -1 \\ -1 & 0 \\ 0 & -1 \end{matrix} \qquad \begin{matrix} -1 & 0 \\ 0 & 1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{matrix}$$

Step 1 of the algorithm which converts all of the current sources to equivalent pressure sources reduces, because there are no pumps in the system, to the simple form of

$$\underline{E}^* = -\check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{I}_T$$

where $\underline{I}_T$ is calculated from $\underline{B}_T \cdot \underline{I}'$ and since the impedances of every branch in the network are the same as those of the network in appendices (7) and (3) (except for the short circuits having zero impedance), $\underline{E}^*$ for each section is given by:

$$\begin{matrix} -220 \\ -280 \end{matrix} \qquad \begin{matrix} -255 \\ -300 \end{matrix}$$

$\underline{J}^{(1)}$ the component of flow due to the equivalent mesh networks is calculated for each subnetwork from:

$$\underline{J}^{(1)} = (\check{\underline{C}}_T \cdot \underline{Z}_T \cdot \underline{C}_T + \underline{Z}_L)^{-1} \cdot \underline{E}^*$$

where $(\breve{C}_T \cdot Z_T \cdot C_T + Z_L)$ for each subnetwork is given by:

$$\begin{matrix} 14 & 6 \\ 6 & 17 \end{matrix} \qquad \begin{matrix} 15 & 6 \\ 6 & 21 \end{matrix}$$

and the inverses by

$$\begin{matrix} 0.08415 & -0.0297 \\ -0.0297 & 0.0693 \end{matrix} \qquad \begin{matrix} 0.0752 & -0.0215 \\ -0.0215 & 0.0536 \end{matrix}$$

$J^{(1)}$ for each subnetwork is therefore given by:

$$\begin{matrix} -10.197 \\ -12.870 \end{matrix} \qquad \begin{matrix} -12.741 \\ -10.646 \end{matrix}$$

Step 3 of the algorithm may be omitted since it calculates the effect for interconnection of any nodal flows in the cut branches. In the case above no node to datum flow occurs in either of the two cut branches.

Step 5 of the algorithm may be combined with step 4, which gives the orthogonal flow vector $J_T^{(1)}$ due to the flows in the equivalent meshes using α_{LT}^{**} which is given by:

$$\begin{matrix} -1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \end{matrix}$$

ie α_{LT}^{**} has the dimensions total number of meshes by number of cut branches and indicates the manner in which the cut branches were originally included in the individual meshes.

I^* is then given by $\breve{\alpha}_{LT}^{**} \cdot J^{(1)}$

$$\begin{matrix} 2.544 \\ 2.224 \end{matrix}$$

The pressure contribution due to interconnection is given in step 5 by:

$$e_T = (\breve{\alpha}_{LT}^{**} (\breve{C} \cdot Z \cdot C)^{-1} \alpha_{LT}^{**} + Y_T)^{-1} \cdot I^*$$

where $\underline{Y}_T$ is the admittance of the cut branches and where

$$(\underline{\alpha}^{**}_{LT}.(\underline{\breve{C}}.\underline{Z}.\underline{C})^{-1}.\underline{\alpha}^{**}_{LT} + \underline{Y}_T)$$

is given by:

0.3594	0.051
0.051	0.2659

and its inverse by:

2.86	-0.548
-0.548	3.866

to give for $\underline{e}'_T$:

6.057

7.203

In step 6 $\underline{e}'_T$ is transformed to the mesh variable $\underline{e}$ from which the second component of flow $\underline{J}^{(2)}$ may be calculated for each mesh of the now connected system by:

$$\underline{J}^{(2)} = (\underline{\breve{C}}.\underline{Z}.\underline{C})^{-1}\ \underline{e}$$

to give for $\underline{J}^{(2)}$:

-0.722

0.6789

0.6106

-0.5174

Individual mesh flows of the connected system are then given by $\underline{J}^{(1)} + \underline{J}^{(2)}$ to give for $\underline{i}'$:

-10.92

-12.19

-12.13

-11.163

These individual mesh flows can be seen to be exactly the same as the mesh flows calculated in appendix (3) by the orthogonal mesh solution method. By further matrix multiplication it can be shown that the node to datum path pressures, calculated using the above mesh flows, will give

the same results for the node to datum pressures of appendices (8) and (7).

APPENDIX (10)

THE LINK AT A TIME METHOD

It has been indicated that if a network consists of tree branches only, then the inverse of the nodal solution matrix given by

$$(\underline{\breve{A}}_T \cdot \underline{Y}_T \cdot \underline{A}_T + \underline{\breve{A}}_L \cdot \underline{Y}_L \cdot \underline{A}_L)^{-1}$$

may be more easily obtained without inversion by performing the simple matrix multiplications to give the compound matrix shown below:

$$(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$$

Branin has proposed (19) an algorithm for using the $(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$ solution matrix for solving the equations of networks which contain link branches. The method is based on the assumption that the link branches of a network are at first ignored and the $(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$ matrix evaluated for the remaining tree branches. Each element of the $(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$ matrix is then updated according to the formula given below

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

for every link branch that has to be added. For instance, the network of appendix (5), if solved by this method, requires the formation of an 8x8 $(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$ matrix. Every element of this matrix is then updated by a value given by the formula above a total of four times. The resulting updated solution matrix should be identical to the inverse of the nodal solution equation formed by conventional methods. It has been experienced however that round-off errors are likely to influence the final values of the elements of the updated $(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$ matrix. To minimise these round-off errors it is suggested that the tree of minimum resistance is chosen for the formation of $(\underline{\breve{B}}_T \cdot \underline{Z}_T \cdot \underline{B}_T)$.

Branin claims that the method would involve less computational effort than the standard procedure for calculating the inverse of the nodal

solution matrix. However the published literature gives no indication that the method has been implemented in a computer program.

A program called LATTNODEFLAN which implements the proposed method has been used to solve fluid network problems and evaluate the claim that the method results in a saving in computational effort. The program is discussed in Chapter (5).

APPENDIX (11)

THE ICL 1905 SERIES MACHINE

The machine for which all of the computer programs of this thesis are written is an ICL 1905 series computer with 32K words of 2 μ sec store (K = 1024, word = 24 bits). The input and output peripherals which have been used are:

1911 card reader:	900 cards per minute
1916 paper tape reader:	1000 characters per second
1933 lineprinter:	1350 lines per minute at 120 characters per line
1925 paper tape punch:	110 characters per second

The backing store which is used for the two diakoptic programs consists of:

- Four magnetic tape decks using 0.5 inch tapes. Information is stored across 7 tracks at a density of 556 characters per inch. The maximum transfer rate is 20,000 characters per second.
- Two exchangeable magnetic disc drives, each drive holding 8,192,000 characters stored on 200 tracks; the transfer rate is 208,000 characters per second.

Jobs to be handled by the University computing staff are normally processed under the control of the Operating System, known as GEORGE 2 (40), which consists of three programs. These three programs input jobs, process jobs, and output jobs simultaneously. The use of this Operating System is known to occupy a large amount of the available fast access store ($\simeq$ 13K), with the result that there is a limit to the size of problem that can be tackled without using the backing store outlined above. In the latter stages of the preparation of this thesis an ICL 1904A at Loughborough University became available for use by the author. This machine, which has 128K of fast access storage, has allowed the

solution of more complex network problems to be attempted.

APPENDIX (12)

LISTINGS OF COMPUTER PROGRAMS

APPENDIX (12.1)

THE PROGRAM HCMESHIN

```
'BEGIN''INTEGER'MESH,BR,N,M,QQ,MAX,PP;
MESH:=READ;
BR:=READ;
MAX:=READ;
'BEGIN''REAL'SUM1,SUM2,ERROR,PI,RHO,MU,ESPI,X;
'ARRAY'L,D,REL[1:BR,1:1],R,RE,PHI,Q1,Q2,Q3,Q4[1:BR];
'INTEGER''ARRAY'C[1:MESH,1:BR],CON[1:MESH,1:MAX],
        NUM[1:MESH];

'PROCEDURE'FINDPHI;
'BEGIN''REAL'K,H;
'REAL''ARRAY'PRESS[1:BR],REQ[1:BR];
H:=RHO*12960000.0*32.2/(4*MU↑2);
K:=RHO/(1811.25*PI↑2);
'FOR' N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
Q4[N]:=D[N,1]↑5/(K*Q1[N]*L[N,1]*PHI[N]);
PRESS[N]:=Q1[N]/Q4[N];
REQ[N]:=SQRT(ABS(PRESS[N]*D[N,1]↑3*H/L[N,1]));
'BEGIN'
'IF'REQ[N]'LE'129.46'THEN'
WRITETEXT('('''('2C6S')'LAMINAR%FLOW%IN%PIPE')');
PRINT(N,3,0);NEWLINE(1);
RE[N]:=(REQ[N]↑2)/8;
'END''ELSE'
'BEGIN'
RE[N]:=-2.5*REQ[N]*LN(REL[N,1]/(D[N,1]*3.7)+1/
(1.13*REQ[N]));
'IF'RE[N]'LE'3000'THEN'
'BEGIN'
WRITETEXT('('''('1C6S')'TRANSITIONAL%FLOW%IN%PIPE')');
NEWLINE(1);PRINT(N,3,0);
'END';
'END';
PHI[N]:=(REQ[N]/RE[N])↑2;
'END';
'END'FINDPHI;
```

```
PI:=3.1421;
ESPI:=READ;
X:=READ;
RHO:=READ;
MU:=READ;
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
C[N,M]:=0;
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'NUM[N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
'BEGIN'CON[N,M]:=READ;
'IF'CON[N,M]'LE'0'THEN''BEGIN'CON[N,M]:=CON[N,M]*(-1);
C[N,CON[N,M]]:=-1;'END''ELSE'C[N,CON[N,M]]:=1;
'END''END';
'FOR'N:=1'STEP'1'UNTIL'BR'DO'Q1[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'L[N,1]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'D[N,1]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'REL[N,1]:=READ;
WRITETEXT('('''('2C6S')' THE%FOLLOWING%ARE%THE%INPUT%DATA')');
WRITETEXT('('''('2C6S')'DENSITY%IS')'); PRINT(RHO,3,3);
WRITETEXT('('LB/CU.FT')');
SPACE(3);
WRITETEXT('('VISCOSITY%IS')'); PRINT(MU,3,3);
WRITETEXT('('LB/FT.HR.')');
SPACE(3);
WRITETEXT('('''('1C6S')'CONVERGENCE%CRITERIA%IS')');
PRINT(ESPI,1,4);
WRITETEXT('('PERCENT%OF%LAST%CALCULATED%VALUE')');
WRITETEXT('('''('2C2S')'PIPE%NO.%%%%LENGTH%FT.%%%%DIAMETER%
FT.%%%%%%%%%ROUGHNESS ')');
NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(4); PRINT(N,3,0);
SPACE(3);PRINT(L[N,1],6,1);SPACE(8);PRINT(D[N,1],1,4);
SPACE(6);PRINT(REL[N,1],1,5);
NEWLINE(1);
'END';
```

```
PAPERTHROW;
QQ:=1;
L4:'IF'QQ=1'THEN''BEGIN''FOR'N:=1'STEP'1'UNTIL'BR'DO'
PHI[N]:=0.002 'END'
'ELSE'FINDPHI;
QQ:=QQ+1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
R[N]:=2*PHI[N]*L[N,1]*RHO/(32.2*D[N,1]'**'5);
Q3[N]:=ABS(Q1[N])'END';
L5:'FOR'N:=1'STEP'1'UNTIL'BR'DO'
Q2[N]:=ABS(Q1[N]);
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'SUM1:=SUM2:=0;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
SUM1:=C[N,CON[N,M]]*SIGN(Q1[CON[N,M]])*R[CON[N,M]]*
(ABS(Q1[CON[N,M]])'**'X)+SUM1;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
SUM2:=X*ABS(C[N,CON[N,M]]*R[CON[N,M]]*(ABS(Q1[CON[N,M]])
'**'(X-1)))+SUM2;
'IF'SUM2#0'THEN''BEGIN'
ERROR:=-SUM1/SUM2;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
Q1[CON[N,M]]:=Q1[CON[N,M]]+C[N,CON[N,M]]*ERROR;
'END''END';
M:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'IF'ABS(Q2[N]-ABS(Q1[N]))/Q2[N]'LE'ESPI'THEN'M:=M+1;
PRINT(M,2,1); NEWLINE(1);
'IF'M#BR'THEN''GOTO'L5;
M:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'IF'ABS(Q3[N]-ABS(Q1[N]))/Q3[N]'LE'ESPI'THEN'M:=M+1;
'IF'M#BR'THEN''GOTO'L4;
PRINT(M,2,0); NEWLINE(1);
```

```
PAPERTHROW;
WRITETEXT('(''('2C6S')'THE%NO.%OF%ITERATIONS%IN%THIS%
CYCLE%IS%')');SPACE(2);PRINT(QQ-1,3,1);
WRITETEXT('(''('2C10S')'BR.NO.%%%%%%FLOW%Q%%%%%%%PHI%%%%
%%%%L%%%%%%%%D%%%%%%%%REL%%%%%%%%%RE.NO.')');
NEWLINE(2);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(8);PRINT(N,3,0);
SPACE(5);PRINT(Q2[N],6,2);
SPACE(2);PRINT(PHI[N],0,5);
SPACE(2);PRINT(L[N,1],6,1);
SPACE(2);PRINT(D[N,1],1,5);
SPACE(2);PRINT(REL[N,1],1,6);
SPACE(2);PRINT(RE[N],5,0);
NEWLINE(1);
'END';
'END''END' OF PROGRAM;
```

APPENDIX (12.2)

THE PROGRAM HCMESHOUT

```
'BEGIN''INTEGER'MESH,BR,N,M,QQ,MAX,PP;
MESH:=READ;
BR:=READ;
MAX:=READ;
'BEGIN''REAL'SUM1,SUM2,ERROR,PI,RHO,MU,ESPI,X;
'ARRAY'L,D,REL[1:BR,1:1],R,RE,PHI,Q1,Q2,Q3,Q4[1:BR];
'INTEGER''ARRAY'C[1:MESH,1:BR],CON[1:MESH,1:MAX],
NUM[1:MESH];

'PROCEDURE'FINDPHI;
'BEGIN''REAL'K,H;
'REAL''ARRAY'PRESS[1:BR],REQ[1:BR];
H:=RHO*12960000.0*32.2/(4*MU↑2);
K:=RHO/(1811.25*PI↑2);
'FOR' N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
Q4[N]:=D[N,1]↑5/(K*Q1[N]*L[N,1]*PHI[N]);
PRESS[N]:=Q1[N]/Q4[N];
REQ[N]:=SQRT(ABS(PRESS[N]*D[N,1]↑3*H/L[N,1]));
'IF'REQ[N]'LE'129.46'THEN'
'BEGIN'
WRITETEXT('('''('2C6S')'LAMINAR%FLOW%IN%PIPE')');
PRINT(N,3,0);NEWLINE(1);
RE[N]:=(REQ[N]↑2)/8;
'END''ELSE'
'BEGIN'
RE[N]:=-2.5*REQ[N]*LN(REL[N,1]/(D[N,1]*3.7)+1/(1.13*REQ[N]));
'IF'RE[N]'LE'3000'THEN'
'BEGIN'
WRITETEXT('('''('1C6S')'TRANSITIONAL%FLOW%IN%PIPE')');
PRINT(N,3,0);NEWLINE(1);
'END';
'END';
PHI[N]:=(REQ[N]/RE[N])↑2;
'END';
'END'FINDPHI;
```

```
PI:=3.141
ESPI:=READ;
X:=READ;
RHO:=READ;
MU:=READ;
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
C[N,M]:=0;
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'NUM[N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
'BEGIN'CON[N,M]:=READ;
'IF'CON[N,M]'LE'0'THEN''BEGIN'CON[N,M]:=CON[N,M]*(-1);
C[N,CON[N,M]]:=-1;'END''ELSE'C[N,CON[N,M]]:=1;
'END';
'END';
'FOR'N:=1'STEP'1'UNTIL'BR'DO'Q1[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'L[N,1]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'D[N,1]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'REL[N,1]:=READ;
WRITETEXT('('''('2C6S')' THE%FOLLOWING%ARE%THE%INPUT%
          DATA')');
WRITETEXT('('''('2C6S')'DENSITY%IS')'); PRINT(RHO,3,3);
WRITETEXT('('LB/CU.FT')');SPACE(3);
WRITETEXT('('VISCOSITY%IS')'); PRINT(MU,3,3);
WRITETEXT('('LB/FT.HR.')');SPACE(3);
WRITETEXT('('''('1C6S')'CONVERGENCE%CRITERIA%IS')');
PRINT(ESPI,1,4);
WRITETEXT('('PERCENT%OF%LAST%CALCULATED%VALUE')');
WRITETEXT('('''('2C2S')'PIPE%NO.%%%%LENGTH%FT.%%%%DIAMETER%
         FT.%%%%%%%%ROUGHNESS ')');
NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(4); PRINT(N,3,0);SPACE(3);PRINT(L[N,1],6,1);
SPACE(8);PRINT(D[N,1],1,4); SPACE(6);PRINT(REL[N,1],1,5);
NEWLINE(1);
'END';
```

```
PAPERTHROW;
QQ:=1;
L4:'IF'QQ=1'THEN''BEGIN''FOR'N:=1'STEP'1'UNTIL'BR'DO'
PHI[N]:=0.05'END'
'ELSE'FINDPHI;
QQ:=QQ+1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
R[N]:=2*PHI[N]*L[N,1]*RHO/(32.2*D[N,1]'**'5);
Q3[N]:=ABS(Q1[N])'END';
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
Q2[N]:=ABS(Q1[N]);
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'SUM1:=SUM2:=0;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
SUM1:=C[N,CON[N,M]]*SIGN(Q1[CON[N,M]])*R[CON[N,M]]*
(ABS(Q1[CON[N,M]])'**'X)+SUM1;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
SUM2:=X*ABS(C[N,CON[N,M]]*R[CON[N,M]]*(ABS(Q1[CON[N,M]])
'**'(X-1)))+SUM2;
'IF'SUM2#0'THEN''BEGIN'
ERROR:=-SUM1/SUM2;
'FOR'M:=1'STEP'1'UNTIL'NUM[N]'DO'
Q1[CON[N,M]]:=Q1[CON[N,M]]+C[N,CON[N,M]]*ERROR;
'END''END';
M:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'IF'ABS(Q2[N]-ABS(Q1[N]))/Q2[N]'LE'ESPI'THEN'M:=M+1;
PRINT(M,2,0); NEWLINE(1);
'IF'M#BR'THEN''GOTO'L4;
PAPERTHROW;
WRITETEXT('(''('2C6S')'THE%NO.%OF%ITERATIONS%IN%THIS%
CYCLE%IS%')');SPACE(2);PRINT(QQ-1,3,0);
NEWLINE(2);
WRITETEXT('(''('2C10S')'BR.NO.%%%%%%FLOW%Q%%%%%%%PHI%%%
%%%%%%L%%%%%%%%%D%%%%%%%%REL%%%%%%%%%RE.NO.')');
NEWLINE(2);
```

```
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'SPACE(8);PRINT(N,3,0);SPACE(5);PRINT(Q2[N],6,2);
SPACE(2);PRINT(PHI[N],0,5);SPACE(2);PRINT(L[N,1],6,1);
SPACE(2);PRINT(D[N,1],1,5);SPACE(2);PRINT(REL[N,1],1,6);
SPACE(2);PRINT(RE[N],5,0);NEWLINE(1);
'END';
'END';
'END'OFHCMESHIN;
```

APPENDIX (12.3)

THE PROGRAM NODEFLAN

```
'BEGIN''INTEGER'N,M,A,B,P,BR,NODE,NU,NL,Q,ND;
      'REAL'RHO,MU,PI,CONVF,K,H,DATUM;
      WRITETEXT('('('2C6S')'THIS%PROGRAM%FINDS%FLOWS%IN%
      PIPE%NETWORKS%BY%THE%CLASSICAL%NODAL%METHOD')');
      NEWLINE(2);
      BR:=READ;
      NODE:=READ;
      ND:=NODE;
'BEGIN''REAL''ARRAY'L,D,REL,PUMP,REQ,RE,PHI,V1,V2,Y,
      PRESS[1:BR],C[1:2*BR],NPOTE,FLONOD[1:NODE],
      OMEGA[1:NODE,1:NODE];
'BEGIN''PROCEDURE'FORMY;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO'
      Y[N]:=D[N]↑5/(K*V1[N]*L[N]*¬HI[N]);
      'PROCEDURE'EVALUATE;
'BEGIN''REAL''ARRAY'ALPHA[1:NODE];
       'PROCEDURE'MYTRIX(ALPHA,NU);
       'VALUE'NU;
       'INTEGER'NU;
       'ARRAY'ALPHA;
'BEGIN''REAL''ARRAY'V[1:NU-1];
       'REAL'Y,PIVOT;
       'INTEGER' I,J,K;
       'FOR'K:=0'STEP'1'UNTIL'NU-1'DO'
'BEGIN'PIVOT:=1.0/ALPHA[1,1];
       'FOR'I:=2'STEP'1'UNTIL'NU'DO'
       V[I-1]:=ALPHA[1,I];
       'FOR'I:=1'STEP'1'UNTIL'NU-1'DO'
'BEGIN'ALPHA[I,NU]:=Y:=-V[I]*PIVOT;
       'FOR'J:=I'STEP'1'UNTIL'NU-1'DO'
       ALPHA[I,J]:=ALPHA[I+1,J+1]+V[J]*Y;
'END';
       ALPHA[NU,NU]:=-PIVOT;
'END';
      'FOR'I:=1'STEP'1'UNTIL'NU'DO'
      'FOR'J:=I'STEP'1'UNTIL'NU'DO'
      ALPHA[I,J]:=ALPHA[J,I]:=-ALPHA[I,J];
'END'MYTRIX;
```

```
        'FOR'M:=1'STEP'1'UNTIL'NODE'DO'
        'FOR'N:=1'STEP'1'UNTIL'NODE'DO'
        OMEGA[M,N]:=0;
        M:=1;
        'FOR'N:=1'STEP'2'UNTIL'2*BR-1'DO'
'BEGIN'A:=ABS(C[N]);B:=ABS(C[N+1]);
        'IF'B'GT'NODE'THEN''GOTO'L6;
        OMEGA[A,B]:=OMEGA[B,A]:=-Y[M];
        OMEGA[B,B]:=OMEGA[B,B]+Y[M];
   L6:OMEGA[A,A]:=OMEGA[A,A]+Y[M];
        M:=M+1;
'END';
        MYTRIX(OMEGA,NODE);
        'FOR'N:=1'STEP'1'UNTIL'NODE'DO'
        ALPHA[N]:=0;
        M:=1;
        'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'A:=ABS(C[M]);B:=ABS(C[M+1]);
        'IF'B'GT'NODE'THEN''GOTO'L7;
        ALPHA[B]:=ALPHA[B]+B*Y[N]*PUMP[N]/C[M+1];
   L7:ALPHA[A]:=ALPHA[A]+A*Y[N]*PUMP[N]/C[M];
        M:=M+2;
'END';
        'FOR'N:=1'STEP'1'UNTIL'NODE'DO'
        ALPHA[N]:=FLONOD[N]-ALPHA[N];
        'FOR'N:=1'STEP'1'UNTIL'NODE'DO'
'BEGIN'NPOTE[N]:=DATUM;
        'FOR'M:=1'STEP'1'UNTIL'NODE'DO'
        NPOTE[N]:=OMEGA[N,M]*ALPHA[M]+NPOTE[N];
'END';
        M:=1;
        'FOR'N:=1'STEP'2'UNTIL'2*BR-1'DO'
'BEGIN'A:=ABS(C[N]);B:=ABS(C[N+1]);
        'IF'B'GT'NODE'THEN''GOTO'L8;
        PRESS[M]:=A*NPOTE[A]/C[N]+B*NPOTE[B]/C[N+1];
        'GOTO'L9;
   L8:PRESS[M]:=A*NPOTE[A]/C[N]+B*DATUM/C[N+1];
   L9:PRESS[M]:=PRESS[M]+PUMP[M];
```

```
      M:=M+1;
'END';
      'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'REQ[N]:=SQRT(ABS(PRESS[N]*D[N]↑3*H/L[N]));
      'IF'REQ[N]'LE'129.46'THEN'
'BEGIN'WRITETEXT('('('('2C6S')'LAMINAR%FLOW%IN%PIPE')');
      PRINT(N,3,0);
      NEWLINE(1);
      RE[N]:=(REQ[N]↑2)/8;
'END''ELSE'
'BEGIN'RE[N]:=-2.5*REQ[N]*LN(REL[N]/(D[N]*3.7)+1/
       (1.13*REQ[N]));
      'IF'RE[N]'LE'3000'THEN'
'BEGIN'WRITETEXT('('('('2C6S')'TRANSITIONAL%FLOW%IN%PIPE')');
      PRINT(N,3,0);
      NEWLINE(1);
'END';
'END';
'END';
      M:=0;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'PHI[N]:=(REQ[N]/RE[N])↑2;
      V1[N]:=RE[N]*PI*D[N]*MU/(4*60*RHO);
      'IF'Q=1'THEN'V2[N]:=1;
      'IF'ABS(V2[N]-V1[N])/V1[N]'LE'CONVF'THEN'
      M:=M+1;
      V2[N]:=V1[N];
'END';
      Q:=Q+1;
PRINT(M,2,1);    NEWLINE(1);
      'IF'M#BR'THEN''GOTO'L1;
'END';

      'PROCEDURE'INPUTS;
'BEGIN'WRITETEXT('('THESE%ARE%THE%INPUT%DATA')');NEWLINE(1);
      WRITETEXT('('DENSITY%OF%FLUID%IS')');PRINT(RHO,3,5);
      WRITETEXT('('LB./CU.FT.')');NEWLINE(1);
```

```
      WRITETEXT('('VISCOSITY%OF%FLUID%IS')');PRINT(MU,3,5);
      WRITETEXT('('LB./FT.HR.')');NEWLINE(1);
      WRITETEXT('('CONVERGENCE%CRITERION%IS')');
      PRINT(CONVF,1,5);
      WRITETEXT('('PERCENT%OF%LAST%CALCULATED%VALUE')');
      WRITETEXT('(''('2C3S')'PIP%NO.%FROM%NODE%%TO%%NODE%
      %%%LENGTH%FT.%%%DIAMETER%FT.')');
      M:=1;
      NEWLINE(1);
      'FOR'N:=1'STEP'2'UNTIL'2*BR'DO'
'BEGIN'PRINT(M,3,0);SPACE(8);
      'IF'C[N]'LT'1'THEN'
'BEGIN'PRINT(C[N],3,0);SPACE(4);PRINT(C[N+1],3,0);
'END'  'ELSE'
'BEGIN'PRINT(C[N+1],3,0);SPACE(4);PRINT(C[N],3,0);
'END';
      SPACE(5);PRINT(L[M],6,3);SPACE(5);PRINT(D[M],2,5);
      NEWLINE(1);
      M:=M+1;
'END';
      PAPERTHROW;
      WRITETEXT('(''('10S')'NODE%NO.%%%%%%%%%INPUT%
      CU.FT./MIN.')');NEWLINE(1);
      'FOR'N:=1'STEP'1'UNTIL'NODE'DO'
'BEGIN'SPACE(10);PRINT(N,3,1);SPACE(10);PRINT(FLONOD[N],5,3);
       NEWLINE(1);
'END';
'END' INPUTS;

      'PROCEDURE'SHOW;
'BEGIN'NEWLINE(10);
      WRITETEXT('(''('2C10')'RESULTS%OF%GIVEN%DATA')');
      WRITETEXT('(''('2C4S')'PIPE%NO.%%FROM%NODE%%TO%%NODE%
      %%%FLOW%CU./FT.%%%%PRESSURE%LB./SQ.FT.%%%%ROUGHNESS
      ')');
      NEWLINE(2);
      M:=1;
      'FOR'N:=1'STEP'2'UNTIL'2*BR'DO'
'BEGIN'PRINT(M,3,0);SPACE(8);
```

```
      'IF'C[N]'LT'1'THEN'
'BEGIN'PRINT(ABS(C[N]),3,0);SPACE(4);PRINT(C[N+1],3,0);
'END' 'ELSE'
'BEGIN'PRINT(ABS(C[N+1]),3,0);SPACE(4);PRINT(C[N],3,0);
'END'; SPACE(5);PRINT(V1[M],5,5);SPACE(6);PRINT(PRESS[M]
       ,5,4);SPACE(8);PRINT(PHI[M],1,5);NEWLINE(1);
      M:=M+1;
'END';
      PAPERTHROW;
      WRITETEXT('(''('10S')'NODE%NO.%%%%%%%%PRESSURE%AT%NO
      DE')');
      NEWLINE(2);
      'FOR'N:=1'STEP'1'UNTIL'NODE'DO'
'BEGIN'SPACE(10);PRINT(N,3,0);SPACE(15);PRINT(NPOTE[N],5,3);
       NEWLINE(1);
'END';
'END'SHOW;

      'COMMENT'THIS IS THE START OF THE PROGRAM PROPER;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO' L[N]:=READ;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO' D[N]:=READ;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO' PUMP[N]:=READ;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO' REL[N]:=READ;
      'FOR'N:=1'STEP'1'UNTIL'2*BR'DO'C[N]:=READ;
      'FOR'N:=1'STEP'1'UNTIL'NODE'DO'FLONOD[N]:=READ;
      'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'V1[N]:=1.0;PHI[N]:=0.002;
'END';
      RHO:=READ;
      MU:=READ;
      PI:=READ;
      CONVF:=READ;
DATUM:=READ;
      K:=RHO/(1811.25*PI↑2);
      H:=RHO*12960000.0*32.2/(4*MU↑2);
      INPUTS;
      Q:=1;
```

```
   L1:FORMY;
      EVALUATE;
      PAPERTHROW;
      WRITETEXT('(''('2C6S')'PROGRAM%TOOK')');PRINT(Q-1,3,1);
      WRITETEXT('('ITERATIONS%TO%REACH%A%SOLUTION')');
      SHOW;
'END';
'END';
'END'OF PROGRAM;
```

APPENDIX (12.4)

THE PROGRAM MESHFLAN

```
'BEGIN'
'INTEGER'N,M,K,BR,ND,MESH,Q,NUM;
'REAL'RHO,MU,PI,X,Y,CONVF;
WRITETEXT('('''('2C10S')'THIS%PROGRAM%IS%BEING%DEVELOPED%TO%
SOLVE%PIPE%NETWORK%FLOW%PROBLEMS%BY%THE%METHOD%OF%MESH%FLOW
%ANALYSIS')');
NEWLINE(1);
BR:=READ;
ND:=READ;
MESH:=BR-ND;
'BEGIN'
'REAL''ARRAY'L,D,REL,PUMP,PRESS,RE,REQ,PHI,V1,V2,V3,DUM1,
Z[1:BR],NOD[1:2*BR],OMEGA[1:MESH,1:MESH],FLONOD[1:ND],
B[1:BR,1:ND],C[1:BR,1:MESH];
'BEGIN'
'PROCEDURE'FORMZ;
'BEGIN'
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
Z[N]:=PHI[N]*L[N]*V1[N]*X/(D[N]↑5);
'END'FORMZ;
'PROCEDURE'EVALUATOR;
'BEGIN'
'REAL''ARRAY'DUM2[1:BR],DUM3,IDASH[1:MESH];
'FOR'M:=1'STEP'1'UNTIL'MESH'DO'
'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
OMEGA[M,K]:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'IF' C[N,M]#0 'THEN'
'BEGIN'
'IF' C[N,K]#0 'THEN'
OMEGA[M,K]:=OMEGA[M,K]+C[N,M]*Z[N]*C[N,K];
'END';
'END';
```

```
'BEGIN'
'PROCEDURE'MYTRIX(ALPHA,NU);
        'VALUE'NU;
        'INTEGER'NU;
        'ARRAY'ALPHA;
'BEGIN''REAL''ARRAY'V[1:NU-1];
        'REAL'Y,PIVOT;
        'INTEGER' I,J,K;
        'FOR'K:=0'STEP'1'UNTIL'NU-1'DO'
'BEGIN'PIVOT:=1.0/ALPHA[1,1];
        'FOR'I:=2'STEP'1'UNTIL'NU'DO'
        V[I-1]:=ALPHA[1,I];
        'FOR'I:=1'STEP'1'UNTIL'NU-1'DO'
'BEGIN'ALPHA[I,NU]:=Y:=-V[I]*PIVOT;
        'FOR'J:=I'STEP'1'UNTIL'NU-1'DO'
        ALPHA[I,J]:=ALPHA[I+1,J+1]+V[J]*Y;
'END';
        ALPHA[NU,NU]:=-PIVOT;
'END';
      'FOR'I:=1'STEP'1'UNTIL'NU'DO'
      'FOR'J:=I'STEP'1'UNTIL'NU'DO'
      ALPHA[I,J]:=ALPHA[J,I]:=-ALPHA[I,J];
'END'MYTRIX;
MYTRIX(OMEGA,MESH);
'FOR' K:=ND+1'STEP'1'UNTIL' BR'DO'
DUM2[K]:=0;
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
DUM2[K]:=DUM1[K]*Z[K];
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
DUM3[N]:=0;
'FOR'K:=1'STEP'1'UNTIL'BR'DO'
'IF' C[K,N]#0 'THEN'
DUM3[N]:=DUM3[N]+C[K,N]*(PUMP[K]-DUM2[K]);
'END';
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
IDASH[N]:=0;
'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
IDASH[N]:=IDASH[N]+OMEGA[N,K]*DUM3[K];
'END';
```

```
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
V2[N]:=0;
'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
'IF'C[N,K]#0 'THEN'
V2[N]:=C[N,K]*IDASH[K]+V2[N];
'END';
'FOR'N:=1'STEP'1'UNTIL'ND'DO'
V2[N]:=V2[N]+DUM1[N];
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PRESS[N]:=V2[N]*Z[N] ;
REQ[N]:=SQRT(ABS(PRESS[N]*D[N]↑3*Y/L[N]));
'IF'REQ[N]'LE'129.46'THEN'
'BEGIN'
WRITETEXT('('('2C6S')'LAMINAR%FLOW%IN%PIPE')');
PRINT(N,3,0);
NEWLINE(1);
RE[N]:=REQ[N]↑2/8;
'END''ELSE'
'BEGIN'
RE[N]:=-2.5*REQ[N]*LN(REL[N]/(D[N]*3.7)+1/(1.13*REQ[N]));
'IF'RE[N]'LE'3000'THEN'
'BEGIN'
WRITETEXT('('('1C6S')'TRANSITIONAL%FLOW%IN%PIPE')');
PRINT(N,3,0);
NEWLINE(1);
'END';
'END';
'END';
M:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PHI[N]:=(REQ[N]/RE[N])↑2;
V1[N]:=RE[N]*PI*D[N]*MU/(4*60*RHO);
'IF'Q=1'THEN'V3[N]:=1.0:
'IF'ABS(V3[N]-V1[N])/V3[N]'LE'CONVF'THEN'M:=M+1;
V3[N]:=V1[N];
'END';
```

```
PRINT(M,3,0);  NEWLINE(1);
Q:=Q+1;
'IF'M#BR'THEN''GOTO'L1'ELSE''GOTO'L2;
'END';
'END'EVALUATOR;

'PROCEDURE'INPUTS;
'BEGIN'
WRITETEXT('('('('4C6S')'THESE%ARE%THE%INPUT%DATA')');
NEWLINE(1);
WRITETEXT('('DENSITY%OF%THE%FLUID%IS')');PRINT(RHO,3,5);
WRITETEXT('('LB/CU.FT.')');
NEWLINE(1);
WRITETEXT('('VISCOSITY%OF%FLUID%IS')');PRINT(MU,3,4);
WRITETEXT('('LB./FT.HR.')');
NEWLINE(1);
WRITETEXT('('CONVERGENCE%CRITERION%IS')');PRINT(CONVF,1,5);
WRITETEXT('('PERECENT%OF%LAST%VALUE')');
NEWLINE(1);
WRITETEXT('('('10S')'PIPE.NO.%%%%%%%%%%%LENGTH.%FT.%
%%%%%%%DIAMETER.%FT.')');
NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(8);PRINT(N,3,0);SPACE(10);PRINT(L[N],6,3);SPACE(10);
PRINT(D[N],2,4);
NEWLINE(1);
'END';
PAPERTHROW;
WRITETEXT('('('10S')'NODE.NO.%%%%%%%%%%INPUT%CU.FT/MIN.')');
NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'ND  'DO'
'BEGIN'
SPACE(10);PRINT(N,3,0);SPACE(10);PRINT(FLONOD[N],5,3);
NEWLINE(1);
'END';
'END'INPUTS;
```

```
'PROCEDURE'OUTPUT;
'BEGIN'
PAPERTHROW;
WRITETEXT('(''('2C10S')'RESULTS%OF%GIVEN%DATA')');
NEWLINE(2);
WRITETEXT('(''('10S')'PIPE%NO.%%%%%%%FLOW%CU.FT./MIN.%
%%%%%%%PRESSURE%%%%%%%%%%%%%%%%ROUGHNESS')');
NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(10);PRINT(N,3,0);SPACE(6);PRINT(V1[N],5,5);SPACE(6);
PRINT(PRESS[N],5,6);SPACE(10);PRINT(PHI[N],5,6);
NEWLINE(1);
'END';
PAPERTHROW;
WRITETEXT('(''('10S')'NODE%NO.%%%%%%%%PRESSURE%AT%NODE')');
NEWLINE(2);
'END'OUTPUT;

'PROCEDURE'SEARCHBT(BT,NOD,ND);
'ARRAY'BT,NOD;
'INTEGER'ND;
'BEGIN'
'INTEGER''ARRAY'NCP[1:ND+1,1:ND+1],PR[1:ND];
'INTEGER' I,J,CHECK,S,A,B;
      'FOR'N:=1'STEP'1'UNTIL'ND+1'DO'
      'FOR'J:=1'STEP'1'UNTIL'ND+1'DO'
      NCP[N,J]:=0;
      K:=1;
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'A:=ABS(NOD[K]);B:=ABS(NOD[K+1]);
       NCP[A,B]:=N*SIGN(NOD[K]);
       NCP[B,A]:=-NCP[A,B];
       K:=K+2;
'END';
```

```
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'CHECK:=0;J:=ND+1;
      'FOR'K:=1'STEP'1'UNTIL'ND'DO'
      PR[K]:=0;
SS3:  I:=1;
SS1:  A:=ABS(NCP[I,J]);
      'IF'A=0'THEN'
'BEGIN'I:=I+1;'IF'I=ND+2'THEN''GOTO'SS2'ELSE''GOTO'SS1;
'END' 'ELSE''IF'(CHECK-A)#0'THEN'
'BEGIN'
      PR[A]:=A/(NCP[I,J]);J:=I;CHECK:=A;'GOTO'SS3;
'END';
      S:=I;
SS5:  'IF'I=ND+1'THEN'
'BEGIN'NCP[J,S]:=NCP[S,J]:=0;'GOTO'SS4;
'END';
SS6:  I:=I+1;
      'IF'NCP[I,J]#0'THEN''GOTO'SS1;
      'IF'I=ND+1'THEN''GOTO'SS5'ELSE''GOTO'SS6;
SS4:  'FOR'K:=1'STEP'1'UNTIL'ND'DO'
      BT[K,J]:=PR[K];
'END';
SS2: 'END' SEARCHBT;

'PROCEDURE'MAKEC(BT,NOD,C,ND);
'ARRAY'BT,NOD,C;
'INTEGER'ND;
'BEGIN''INTEGER'N,M,K,A,B;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
      'FOR'M:=1'STEP'1'UNTIL'BR'DO'
      'IF'M=ND+N'THEN'C[M,N]:=1'ELSE'C[M,N]:=0;
      M:=2*ND+1;
      'FOR' N:=1'STEP'1'UNTIL' MESH'DO'
'BEGIN'A:=ABS(NOD[M]);B:=ABS(NOD[M+1]);
      'FOR'K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
      'IF'B>ND'THEN''GOTO'SS11;
```

```
      C[K,N]:=C[K,N]+SIGN(NOD[M+1])*BT[K,B];
SS11: C[K,N]:=C[K,N]+SIGN(NOD[M])*BT[K,A];
      C[K,N]:=-C[K,N];
'END';M:=M+2;
'END';
'END'MAKEC;

Q:=1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' L[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' D[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' PUMP[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' REL [N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'2*BR'DO' NOD[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'ND'DO' FLONOD[N]:=READ;
       SEARCHBT(B,NOD,ND);
       MAKEC(B,NOD,C,ND);
'FOR' K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
DUM1[K]:=0;
'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'IF'B[K,N]#0 'THEN'
DUM1[K]:=DUM1[K]+B[K,N]*FLONOD[N];
'END';
'FOR'K:=ND+1'STEP'1'UNTIL'BR'DO'
DUM1[K]:=1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PHI[N]:=0.002;
V1[N]:= DUM1[N]+1;
'END';
RHO:=READ;
MU:=READ;
PI:=READ;
CONVF:=READ;
X:=RHO/(1811.25*PI↑2); Y:=RHO*12960000.0*32.2/(4*MU↑2);
INPUTS;
```

```
L1:FORMZ;
    EVALUATOR;
L2:OUTPUT;
'END';
'END';
'END'MESHFLAN;
```

APPENDIX (12.5)

THE PROGRAM LATTNODEFLAN

```
'BEGIN'
WRITETEXT('('THIS%PROGRAM%IS%A%VARIATION%ON%THE%CLASSICAL%
THEME% FOR%SOLVING%PIPE%FLOWS%USING%THE%LINK%AT%A%TIME%
METHOD')');
'COMMENT' NOW READ IN THE NO OF BRANCHES AND TREE BRANCHES
 THIS ALLOWS THE PROGRAM TO SET UP ITS ARRAYS;
'BEGIN'
'INTEGER'BR,TBR,A,B,N,M,  NU,NL,Q,K,J;
BR:=READ;
TBR:=READ;
'BEGIN''REAL''ARRAY'L,D,REL,PHI,PUMP,V1,V2,V3,Z,BRPOTE,RE,
REQ[1:BR],FLONOD,NPOTE[1:TBR];
'INTEGER''ARRAY'AT,BT[1:TBR,1:TBR],C[1:2*BR];
'COMMENT'NOW SET UP CONSTANTS TO BE USED THEN GOTO SUBR TO
 READ DATA;
'BEGIN''REAL'RHO,MU,PI,CONVF,X,Y;
'BEGIN'
'PROCEDURE'READDATA;
'BEGIN'
'COMMENT'SET UP INITIAL GUESSES OF V1 AND PHI;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
V1[N]:=1.0;
PHI[N]:=0.05;
'END';
'FOR'N:=1'STEP'1'UNTIL'BR'DO' L[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' D[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' PUMP[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' REL[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'2*BR'DO'C[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'TBR'DO'FLONOD[N]:=READ;
RHO:=READ;
MU:=READ;
PI:=READ;
CONVF:=READ;
'END'READDATA;
```

```
'PROCEDURE'INPUTS;
'BEGIN'
WRITETEXT('('('1C3S')'THESE%ARE%THE%INPUT%DATA')');
NEWLINE(1);
WRITETEXT('('DENSITY%OF%THE%FLUID%IS')');
PRINT(RHO,3,5);
WRITETEXT('('LB/CU.FT.')');
NEWLINE(1);
WRITETEXT('('VISCOSITY%OF%THE%FLUID%IS')');
PRINT(MU,3,5);
WRITETEXT('('LB/FT.HR.')');
NEWLINE(1);
WRITETEXT('('CONVERGENCE%CRITERION%IS')');
PRINT(CONVF,1,5);
NEWLINE(1);
WRITETEXT('('('10S')'PIPE%NO.%%%%%%%%%LENGTH.%FT.%%%%%%%
DIAMETER%FT.')');NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(8);PRINT(N,3,0);SPACE(10);PRINT(L[N],6,3);SPACE(10);
PRINT(D[N],2,5);
NEWLINE(1);
'END';
'END'INPUTS;
      'PROCEDURE'SETUPB;
'BEGIN''REAL''ARRAY'INV[1:TBR,1:2*TBR];
      'INTEGER''ARRAY'W,T[1:TBR];
      'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
      'FOR'J:=1'STEP'1'UNTIL'2*TBR'DO'
      INV[N,J]:=0;
      M:=1;J:=1;
      'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
'BEGIN'
      A:=ABS(C[M]);B:=ABS(C[M+1]);
      'IF'B'GT'TBR'THEN''GOTO'S11;
      INV[N,B]:=B/C[M+1];
 S11:INV[N,A]:=A/C[M];
      INV[N,TBR+J]:=1;
      M:=M+2;J:=J+1;
```

```
'END';
      'FOR'K:=1'STEP'1'UNTIL'TBR'DO'
'BEGIN'W[K]:=0;
      'FOR'J:=2*TBR'STEP'-1'UNTIL'K'DO'
'BEGIN'
      'IF'INV[K,K]=0'THEN'W[K]:=K'ELSE''GOTO'S12;
      'FOR'N:=K'STEP'1'UNTIL'TBR'DO'
'BEGIN'
      'IF'INV[N,K]=0'THEN''GOTO'S13;
      T[K]:=N;
      'FOR'M:=K'STEP'1'UNTIL'2*TBR'DO'
'BEGIN'
      A:=INV[K,M];INV[K,M]:=INV[N,M];INV[N,M]:=A;
'END';
S13: 'GOTO'S12;
'END';
S12:  'IF'INV[K,J]#0'THEN'INV[K,J]:=INV[K,J]/INV[K,K];
'END';
      'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
'BEGIN'
      'IF'N=K'THEN''GOTO'S14;
      'FOR'J:=2*TBR'STEP'-1'UNTIL'K'DO'
'BEGIN'
      'IF'INV[N,K]#0'OR'INV[K,J]#0'THEN'
      INV[N,J]:=INV[N,J]-INV[N,K]*INV[K,J];
'END';
S14: 'END';
'END';
      'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
      'FOR'J:=1'STEP'1'UNTIL'TBR'DO'
      BT[J,N]:=-INV[N,TBR+J];
'END'SETUPB;
'PROCEDURE'ZANDY;
'BEGIN'
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
Z[N]:=X*V1[N]*L[N]*PHI[N]/(D[N]↑5);
'END'ZANDY;
```

```
'PROCEDURE'EVALUATOR;
'BEGIN'
'REAL''ARRAY'DUM1[1:BR],DUM2[1:TBR];
'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
DUM2[N]:=0;
M:=1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'IF' PUMP[N]#0 'THEN'
'BEGIN'
A:=ABS(C[M]);B:=ABS(C[M+1]);
'IF'B'LE'TBR'THEN'
      DUM2[B]:=DUM2[B]+B*PUMP[N]/(Z[N]*C[M+1]);
      DUM2[A]:=DUM2[A]+A*PUMP[N]/(Z[N]*C[M]);
    M:=M+2;
'END' 'ELSE' M:=M+2;
'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
DUM2[N]:=FLONOD[N]-DUM2[N];
'BEGIN'
'PROCEDURE'TRICKS;
'BEGIN''REAL''ARRAY'DUM3,DUM[1:TBR,1:TBR];
'REAL'DUM;
'FOR'M:=1'STEP'1'UNTIL'TBR'DO'
'FOR'K:=1'STEP'1'UNTIL'TBR'DO'
'BEGIN'
DUM4[M,K]:=0;
'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
      'IF' BT[N,M]#0 'THEN'
'BEGIN'
'IF' BT[N,K]#0 'THEN'
DUM4[M,K]:=DUM4[M,K]+BT[N,M]*Z[N]*BT[N,K];
'END';
'END';
M:=TBR+1;
'FOR'N:=2*TBR+1'STEP'2'UNTIL'2*BR'DO'
'BEGIN'
A:=ABS(C[N]);
B:=ABS(C[N+1]);
'IF'B>TBR'THEN' DUM:=DUM4[A,A]+Z[M]   'ELSE'
DUM:=DUM4[A,A]+DUM4[B,B]-DUM4[A,B]-DUM4[B,A]+Z[M];
'FOR'K:=1'STEP'1'UNTIL'TBR'DO'
```

```
'FOR'J:=1'STEP'1'UNTIL'TBR'DO'
'IF'B>TBR'THEN' DUM3[K,J]:=DUM4[K,J]-(DUM4[K,A]*DUM4[A,J])
/DUM'ELSE'DUM3[K,J]:=DUM4[K,J]-((DUM4[K,A]-DUM4[K,B])*
(DUM4[A,J]-DUM4[B,J])/DUM);M:=M+1;
'FOR'K:=1'STEP'1'UNTIL'TBR'DO'
'FOR'J:=1'STEP'1'UNTIL'TBR'DO'
DUM4[K,J]:=DUM3[K,J];
'END';
'FOR'N:=1'STEP'1'UNTIL'TBR'DO'
'BEGIN'
NPOTE[N]:=0;
'FOR'M:=1'STEP'1'UNTIL'TBR'DO'
'IF' DUM2[M]#0 'THEN'
NPOTE[N]:=DUM4[N,M]*DUM2[M]+NPOTE[N];
'END';
M:=1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
A:=ABS(C[M]);
B:=ABS(C[M+1]);
'IF'B'LE'TBR'THEN'
BRPOTE[N]:=A*NPOTE[A]/C[M]+B*NPOTE[B]/C[M+1]  'ELSE'
     BRPOTE[N]:=A*NPOTE[A]/C[M];
     M:=M+2;
'END';
'END'TRICKS;
TRICKS;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
BRPOTE[N]:=BRPOTE[N]+PUMP[N];
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
REQ[N]:=SQRT(ABS(BRPOTE[N]*D[N]↑3*Y/L[N]));
'IF'REQ[N]'LE'129.46'THEN'
'BEGIN'
WRITETEXT('('('2C6S')'LAMINAR%FLOW%IN%PIPE')');
PRINT(N,3,0);
NEWLINE(1);
RE[N]:=REQ[N]↑2/8;
'END''ELSE'
'BEGIN'
```

```
RE[N]:=-2.5*REQ[N]*LN(REL[N]/(D[N]*3.7)+1/(1.13*REQ[N]));
'IF'RE[N]'LE'3000'THEN'
'BEGIN'
WRITETEXT('(''('1C6S')'TRANSITIONAL%FLOW%IN%PIPE')');
PRINT(N,3,0);
NEWLINE(1);
'END';
'END';
'END';
M:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PHI[N]:=(REQ[N]/RE[N])↑2;
V1[N]:=RE[N]*PI*D[N]*MU/(4*60*RHO);
'IF'Q=1'THEN'V3[N]:=1;
     'IF' ABS(V3[N]-V1[N])/V3[N] 'LE' CONVF 'THEN'
M:=M+1;
V3[N]:=V1[N];
'END';
Q:=Q+1;
PRINT(M,2,1);    NEWLINE(1);
'IF'M#BR'THEN''GOTO'L1
'ELSE''GOTO'L4;
'END';
'END'EVALUATOR;

'PROCEDURE'SHOW;
'BEGIN'
PAPERTHROW;
WRITETEXT('(''('2C4S')'NO.%OF%ITERATIONS%PROGRAM%TOOK%FOR%
SOLUTION')');
PRINT(Q-1,3,0);
WRITETEXT('(''('2C10S')'RESULTS%OF%GIVEN%DATA')');
NEWLINE(1);
WRITETEXT('(''('10S')'PIPE%NO.%%%%%%%FLOW%CU.FT./MIN.
%%%%%%%PRESSURE%%%%%%ROUGHNESS')');
NEWLINE(2);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(10); PRINT(N,3,0);SPACE(6);PRINT(V1[N],5,5);
SPACE(6);PRINT(BRPOTE[N],5,6); SPACE(10);PRINT(PHI[N],5,8);
```

```
NEWLINE(1);
'END';
'END'SHOW;
'COMMENT' START OF ITERATION CYCLES ;
READDATA;
'COMMENT' GOTO A SUBR TO PRINT OUT DATA READ IN AS A CHECK;
INPUTS;
X:=RHO/(1811.25*PI↑2);
Y:=RHO*12960000.0*32.2/(4*MU↑2);
'COMMENT'NOW SET UP THE B MATRIX FROM A;
SETUPB;
Q:=1;
L1:ZANDY;
L2:EVALUATOR;
L4:SHOW;
L20:'END';
'END';
'END';
'END';
'END';
```

APPENDIX (12.6)

THE PROGRAM NPSMESHFLAN

```
'BEGIN'
'INTEGER'N,M,K,BR,ND,MESH,NP,Q,NUM;
'REAL'RHO,MU,PI,X,Y,CONVF;
BR:=READ;
ND:=READ;
MESH:=BR-ND;
'BEGIN' 'REAL' 'ARRAY' NODPRES[1:ND];
        NP:=0;
       'FOR'N:=1'STEP'1'UNTIL' ND'DO'
' BEGIN'NODPRES[N]:=READ;
       'IF' NODPRES[N]#0'THEN'
'BEGIN' NP:=NP+1;
        MESH:=MESH+1;
'END';
'END';
'BEGIN'
'REAL''ARRAY'L,D,REL,PUMP,PRESS,RE,REQ,PHI,V1,V2,V3,DUM1,
      Z[1:BR+NP],NOD[1:2*BR+2*NP],OMEGA[1:MESH,1:MESH],
      FLONOD[1:ND],B[1:BR+NP,1:ND],C[1:BR+NP,1:MESH];
'BEGIN'

'PROCEDURE'FORMZ;
'BEGIN'
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
Z[N]:=PHI[N]*L[N]*V1[N]*X/(D[N]↑5);
'END'FORMZ;

'PROCEDURE'EVALUATOR;
'BEGIN'
'REAL''ARRAY'DUM2[1:BR+NP],DUM3,IDASH[1:MESH];
'FOR'M:=1'STEP'1'UNTIL'MESH'DO'
'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
OMEGA[M,K]:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
```

```
      'IF' C[N,M]#0 'THEN'
'BEGIN'
      'IF' C[N,K]#0 'THEN'
     OMEGA[M,K]:=OMEGA[M,K]+C[N,M]*Z[N]*C[N,K];
'END';
'END';
'BEGIN'

        'PROCEDURE'MYTRIX(ALPHA,NU);
        'VALUE'NU;
        'INTEGER'NU;
        'ARRAY'ALPHA;
'BEGIN''REAL''ARRAY'V[1:NU-1];
        'REAL'Y,PIVOT;
        'INTEGER' I,J,K;
        'FOR'K:=0'STEP'1'UNTIL'NU-1'DO'
'BEGIN'PIVOT:=1.0/ALPHA[1,1];
        'FOR'I:=2'STEP'1'UNTIL'NU'DO'
        V[I-1]:=ALPHA[1,I];
        'FOR'I:=1'STEP'1'UNTIL'NU-1'DO'
'BEGIN'ALPHA[I,NU]:=Y:=-V[I]*PIVOT;
        'FOR'J:=I'STEP'1'UNTIL'NU-1'DO'
        ALPHA[I,J]:=ALPHA[I+1,J+1]+V[J]*Y;
'END';
        ALPHA[NU,NU]:=-PIVOT;
'END';
      'FOR'I:=1'STEP'1'UNTIL'NU'DO'
      'FOR'J:=I'STEP'1'UNTIL'NU'DO'
      ALPHA[I,J]:=ALPHA[J,I]:=-ALPHA[I,J];
'END'MYTRIX;

MYTRIX(OMEGA,MESH);
'FOR' K:=ND+1'STEP'1'UNTIL' BR+NP'DO'
DUM2[K]:=0;
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
DUM2[K]:=DUM1[K]*Z[K];
```

```
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
DUM3[N]:=0;
'FOR'K:=1'STEP'1'UNTIL'BR+NP'DO'
'IF' C[K,N]#0 'THEN'
DUM3[N]:=DUM3[N]+C[K,N]*(PUMP[K]-DUM2[K]);
'END';
'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
IDASH[N]:=0;
'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
IDASH[N]:=IDASH[N]+OMEGA[N,K]*DUM3[K];
'END';
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
V2[N]:=0;
'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
'IF'C[N,K]#0 'THEN'
V2[N]:=C[N,K]*IDASH[K]+V2[N];
'END';
'FOR'N:=1'STEP'1'UNTIL'ND'DO'
V2[N]:=V2[N]+DUM1[N];
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PRESS[N]:=V2[N]*Z[N] ;
REQ[N]:=SQRT(ABS(PRESS[N]*D[N]↑3*Y/L[N]));
'IF'REQ[N]'LE'129.46'THEN'
'BEGIN'
WRITETEXT('('''('2C6S')'LAMINAR%FLOW%IN%PIPE')');
PRINT(N,3,0);NEWLINE(1);
RE[N]:=REQ[N]↑2/8;
'END''ELSE'
'BEGIN'
RE[N]:=-2.5*REQ[N]*LN(REL[N]/(D[N]*3.7)+1/(1.13*REQ[N]));
'IF'RE[N]'LE'3000'THEN'
'BEGIN'
WRITETEXT('('''('1C6S')'TRANSITIONAL%FLOW%IN%PIPE')');
```

```
PRINT(N,3,0);NEWLINE(1);
'END';
'END';
'END';
M:=0;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PHI[N]:=(REQ[N]/RE[N])↑2;
V1[N]:=RE[N]*PI*D[N]*MU/(4*60*RHO);
'IF'Q=1'THEN'V3[N]:=1.0;
'IF'ABS(V3[N]-V1[N])/V3[N]'LE'CONVF'THEN'M:=M+1;
V3[N]:=V1[N];
'END';
PRINT(M,3,1);  NEWLINE(1);
Q:=Q+1;
'IF'M#BR'THEN''GOTO'L1'ELSE''GOTO'L2;
'END';
'END'EVALUATOR;

'PROCEDURE'INPUTS;
'BEGIN'
WRITETEXT('('('4C6S')'THESE%ARE%THE%INPUT%DATA')');
NEWLINE(1);
WRITETEXT('('DENSITY%OF%THE%FLUID%IS')');PRINT(RHO,3,5);
WRITETEXT('('LB/CU.FT.')');NEWLINE(1);
WRITETEXT('('VISCOSITY%OF%FLUID%IS')');PRINT(MU,3,4);
WRITETEXT('('LB./FT.HR.')');NEWLINE(1);
WRITETEXT('('CONVERGENCE%CRITERION%IS')');PRINT(CONVF,1,5);
WRITETEXT('('PERECENT%OF%LAST%VALUE')');NEWLINE(1);
WRITETEXT('('('10S')'PIPE.NO.%%%%%%%%%%%LENGTH.%FT.%%%%
%%%%DIAMETER.%FT.')');NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(8);PRINT(N,3,0);SPACE(10);PRINT(L[N],6,3);SPACE(10);
PRINT(D[N],2,4);NEWLINE(1);
'END';
PAPERTHROW;
WRITETEXT('('('10S')'NODE.NO.%%%%%%%%%INPUT%
CU.FT/MIN.')');NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'ND  'DO'
```

```
'BEGIN'
SPACE(10);PRINT(N,3,0);SPACE(10);PRINT(FLONOD[N],5,3);
NEWLINE(1);
'END';
'END'INPUTS;

'PROCEDURE'OUTPUT;
'BEGIN'
PAPERTHROW;
WRITETEXT('('''('2C10S')'RESULTS%OF%GIVEN%DATA')');
NEWLINE(2);
WRITETEXT('('''('10S')'PIPE%NO.%%%%%%%FLOW%CU.FT./MIN.%%%%%
%%%PRESSURE%%%%%%%%%%%%%%ROUGHNESS')');
NEWLINE(1);
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(10);PRINT(N,3,0);SPACE(6);PRINT(V1[N],5,5);SPACE(6);
PRINT(PRESS[N],5,6);SPACE(10);PRINT(PHI[N],5,6);
NEWLINE(1);
'END';
PAPERTHROW;
WRITETEXT('('''('10S')'NODE%NO.%%%%%%%%PRESSURE%AT%NODE')');
NEWLINE(2);
'FOR'N:=1'STEP'1'UNTIL'ND  'DO'
'BEGIN'
PRINT(NODPRES[N],6,2);NEWLINE(1);
'END';
'END'OUTPUT;

'PROCEDURE'SEARCHBT(BT,NOD,ND);
'ARRAY'BT,NOD;
'INTEGER'ND;
'BEGIN'
'INTEGER''ARRAY'NCP[1:ND+1,1:ND+1],PR[1:ND];
'INTEGER' I,J,CHECK,S,A,B;
      'FOR'N:=1'STEP'1'UNTIL'ND+1'DO'
      'FOR'J:=1'STEP'1'UNTIL'ND+1'DO'
      NCP[N,J]:=0;
```

```
      K:=1;
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'A:=ABS(NOD[K]);B:=ABS(NOD[K+1]);
       NCP[A,B]:=N*SIGN(NOD[K]);NCP[B,A]:=-NCP[A,B];
       K:=K+2;
'END';
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'CHECK:=0;J:=ND+1;
      'FOR'K:=1'STEP'1'UNTIL'ND'DO'
      PR[K]:=0;
SS3:  I:=1;
SS1:  A:=ABS(NCP[I,J]);
      'IF'A=0'THEN'
'BEGIN'I:=I+1;'IF'I=ND+2'THEN''GOTO'SS2'ELSE''GOTO'SS1;
'END' 'ELSE''IF'(CHECK-A)#0'THEN'
'BEGIN'
      PR[A]:=A/(NCP[I,J]);J:=I;CHECK:=A;'GOTO'SS3;
'END';
      S:=I;
SS5:  'IF'I=ND+1'THEN'
'BEGIN'NCP[J,S]:=NCP[S,J]:=0;'GOTO'SS4;
'END';
SS6:  I:=I+1;
      'IF'NCP[I,J]#0'THEN''GOTO'SS1;
      'IF'I=ND+1'THEN''GOTO'SS5'ELSE''GOTO'SS6;
SS4:  'FOR'K:=1'STEP'1'UNTIL'ND'DO'
      BT[K,J]:=PR[K];
'END';
SS2: 'END' SEARCHBT;

'PROCEDURE'MAKEC(BT,NOD,C,ND);
'ARRAY'BT,NOD,C;
'INTEGER'ND;
'BEGIN''INTEGER'N,M,K,A,B;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
      'FOR'M:=1'STEP'1'UNTIL'BR+NP'DO'
      'IF'M=ND+N'THEN'C[M,N]:=1'ELSE'C[M,N]:=0;
```

```
      M:=2*ND+1;
      'FOR' N:=1'STEP'1'UNTIL' MESH'DO'
'BEGIN'A:=ABS(NOD[M]);B:=ABS(NOD[M+1]);
      'FOR'K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
      'IF'B>ND'THEN''GOTO'SS11;
      C[K,N]:=C[K,N]+SIGN(NOD[M+1])*BT[K,B];
SS11: C[K,N]:=C[K,N]+SIGN(NOD[M])*BT[K,A];
      C[K,N]:=-C[K,N];
'END';M:=M+2;
'END';
'END'MAKEC;

Q:=1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' L[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' D[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' PUMP[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'BR'DO' REL [N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'2*BR'DO' NOD[N]:=READ;
'FOR'N:=1'STEP'1'UNTIL'ND  'DO' FLONOD[N]:=READ;
M:=K:=1;
      'FOR'N:=1'STEP'1'UNTIL' ND'DO'
'BEGIN' 'IF' NODPRES[N]#0'THEN'
'BEGIN' NOD[2*BR+M]:=-N;
      NOD[2*BR+1+M]:=ND+1;
      PUMP[BR+K]:=NODPRES[N];
      M:=M+2;
      K:=K+1;
'END';
'END';
SEARCHBT(B,NOD,ND);
MAKEC(B,NOD,C,ND);
'FOR' K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
DUM1[K]:=0;
'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'IF'B[K,N]#0 'THEN'
```

```
DUM1[K]:=DUM1[K]+B[K,N]*FLONOD[N];
'END';
'FOR'K:=ND+1'STEP'1'UNTIL'BR'DO'
DUM1[K]:=1;
'FOR'N:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PHI[N]:=0.002;
V1[N]:= DUM1[N]+1;
'END';
RHO:=READ;
MU:=READ;
PI:=READ;
CONVF:=READ;
X:=RHO/(1811.25*PI↑2);
Y:=RHO*12960000.0*32.2/(4*MU↑2);
INPUTS;
L1:FORMZ;
EVALUATOR;
L2:OUTPUT;
'END';
'END';
'END';
'END'OF PROGRAM;
```

APPENDIX (12.7)

THE PROGRAM NODEDIAK

```
'BEGIN''COMMENT'START OF NODEDIAK;
'REAL' RHO,MU,PI,GR,LIMIT,SINGLE;
'INTEGER'CUTSEC,CUTBR,TOTENODE,N,NN,M,MM,Q,QQ,C,CC,CCC,A,B,
        P,COUNT,DECIDE,AA,BB,BR,ND,K,DD,NL,NU,L,Z,ZZ,F;
RHO:=READ;
MU:=READ;
PI:=READ;
GR:=READ;
LIMIT:=READ;
CUTSEC:=READ;
CUTBR:=READ;
TOTENODE:=READ;
'BEGIN'

'PROCEDURE'WORKSTORE(N,A,B);
'VALUE'N,B;
'INTEGER'N,B;
'STRING'A;
'EXTERNAL';

'PROCEDURE'PUTARRAY(N,A,B);
'VALUE'N;
'INTEGER'N,A;
'ARRAY'B;
'EXTERNAL';

'PROCEDURE'GETARRAY(N,A,B);
'VALUE'N;
'INTEGER'N,A;
'ARRAY'B;
'EXTERNAL';

'PROCEDURE'MYTRIX(ALPHA,NU);
      'VALUE'NU;
      'INTEGER'NU;
      'ARRAY'ALPHA;
```

```
'BEGIN''REAL''ARRAY'V[1:NU-1];
        'REAL'Y,PIVOT;
        'INTEGER'I,J,K;
        'FOR'K:=0'STEP'1'UNTIL'NU-1'DO'
'BEGIN'PIVOT:=1.0/ALPHA[1,1];
        'FOR'I:=2'STEP'1'UNTIL'NU'DO'
        V[I-1]:=ALPHA[1,I];
        'FOR'I:=1'STEP'1'UNTIL'NU-1'DO'
'BEGIN'ALPHA[I,NU]:=Y:=-V[I]*PIVOT;
        'FOR'J:=I'STEP'1'UNTIL'NU-1'DO'
        ALPHA[I,J]:=ALPHA[I+1,J+1]+V[J]*Y;
'END';ALPHA[NU,NU]:=-PIVOT;
'END';'FOR'I:=1'STEP'1'UNTIL'NU'DO'
      'FOR'J:=I'STEP'1'UNTIL'NU'DO'
      ALPHA[I,J]:=ALPHA[J,I]:=-ALPHA[I,J];
'END'MYTRIX;

WORKSTORE(1,('('ED')'),2500 );
WORKSTORE(2,('('ED')'),2500 );
WORKSTORE(3,('('ED')'),2500 );
WORKSTORE(4,('('ED')'),2500 );
'BEGIN'
'REAL''ARRAY'MBR,MND[1:CUTSEC],EDASH,EDASHA[1:TOTENODE],
             YB[1:CUTBR,1:CUTBR],CUTCON[1:2*CUTBR],ZED,PHIC,
             VC1,VC2,DC,LC,RELC,PUMPC[1:CUTBR],TEST2[1:TOT
             ENODE];
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'DC[M]:=READ;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'LC[M]:=READ;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'PUMPC[M]:=READ;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'RELC[M]:=READ;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'VC1[M]:=1.0;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'PHIC[M]:=0.002;
'FOR'M:=1'STEP'1'UNTIL'2*CUTBR'DO'
CUTCON[M]:=READ;
'FOR'N:=1'STEP'1'UNTIL'CUTSEC'DO'
'BEGIN'
```

```
MBR[N]:=READ;
MND[N]:=READ;
'END';
COUNT:=1;DECIDE:=0;
L7:CCC:=CC:=QQ:=C:=DD:=Z:=ZZ:=1;AA:=BB:=F:=0;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
YB[M,N]:=0.0;
N:=1;
'FOR'L:=1'STEP'1'UNTIL'CUTSEC'DO'
'BEGIN'F:=F+1;
BR:=MBR[L];
ND:=MND[L];
'BEGIN''REAL''ARRAY'FLONOD[1:ND,1:1],PUMP[1:BR,1:1];
'IF'COUNT'LE'1'THEN'
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
FLONOD[M,N]:=READ;
'BEGIN'

'PROCEDURE'CALCULATE;
'BEGIN''REAL''ARRAY'L,D,PHI,V1,V2,REL,DELTP,Y[1:BR,1:1],
                    NOD[1:2*BR,1:1],OMEGA[1:ND,1:ND];
'IF'COUNT'LE'1'THEN'
'BEGIN'
'FOR'M:=1'STEP'1'UNTIL'BR'DO'D[M,N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'L[M,N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'PUMP[M,N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'REL[M,N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'2*BR'DO'NOD[M,N]:=READ;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
V1[M,N]:=1.0;PHI[M,N]:=0.002;
'END';
PUTARRAY(2,ZZ,D);
PUTARRAY(2,ZZ,L);
PUTARRAY(2,ZZ,PUMP);
PUTARRAY(2,ZZ,REL);
PUTARRAY(2,ZZ,FLONOD);
PUTARRAY(2,ZZ,NOD);
'END''ELSE'
```

```
'BEGIN'
GETARRAY(2,ZZ,D);
GETARRAY(2,ZZ,L);
GETARRAY(2,ZZ,PUMP);
GETARRAY(2,ZZ,REL);
GETARRAY(2,ZZ,FLONOD);
GETARRAY(2,ZZ,NOD);
'END';
'BEGIN'

'PROCEDURE'PIPEDATA;
'BEGIN'
WRITETEXT('('('2C')'DATA%FOR%CUTSECTION%NO.')');
PRINT(F,3,0);
WRITETEXT('('('4C')'PIPE%NO.%FROM%NODE%TO%NODE%%%LENGTH%
FT.%DIAMETER%FT.%REL/R.%FT.')');NEWLINE(2);
K:=1;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
PRINT(M,2,0);SPACE(7);'IF'NOD[K,N]'LT'1'THEN'
'BEGIN'PRINT(ABS(NOD[K,N]),3,0);SPACE(5);
PRINT(NOD[K+1,N],3,0);
'END''ELSE'
'BEGIN'PRINT(ABS(NOD[K+1,N]),3,0);SPACE(5);
PRINT(NOD[K,N],3,0);
'END';
SPACE( 4);PRINT(L[M,N],5,2);SPACE(3);PRINT(D[M,N],1,4);
SPACE(3);PRINT(REL[M,N],1,5);NEWLINE(1);K:=K+2;
'END'
'END'PIPEDATA;

'PROCEDURE'RESULT;
'BEGIN'
PAPERTHROW;
WRITETEXT('('('2C6S')'%%RESULTS%FOR%SECTION%NO.')');
PRINT(F,2,1);NEWLINE(4);
WRITETEXT('('%%%%%PIPE%NO.%%%%NODE%%%TO%%%NODE%%%%PHI%%%%
%%%FLOW%CU.FT./MIN.')');

NEWLINE(1);
```

```
K:=1;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
SPACE(6);PRINT(M,2,0);SPACE(4);'IF'NOD[K,N]'LT'1'THEN'
'BEGIN'PRINT(ABS(NOD[K,N]),3,0);SPACE(8);
PRINT(NOD[K+1,N],3,0);
'END''ELSE'
'BEGIN'PRINT(ABS(NOD[K+1,N]),3,0);SPACE(8);
PRINT(NOD[K,N],3,0);
'END';PRINT(PHI[M,N],1,5);SPACE(5);PRINT(V2[M,N],5,3);
NEWLINE(1);
K:=K+2;
'END';
BB:=AA-ND+1;
WRITETEXT('(''('4C1S')'NODE%NO.%%%%%%%%PRESSURE')');
NEWLINE(1);
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
SPACE(2);PRINT(M,2,0);SPACE(9);PRINT(EDASH[BB],3,2);
NEWLINE(1);
BB:=BB+1;
'END';
'END'RESULT;

'PROCEDURE'FINDFLOW;
'BEGIN''REAL''ARRAY'REQ,RE[1:BR,1:1];
K:=1;
'FOR'M:=1'STEP'2'UNTIL'2*BR-1'DO'
'BEGIN'
A:=ABS(NOD[M,N]);
B:=ABS(NOD[M+1,N]);
'IF'B'LE'ND'THEN'
DELTP[K,N]:=PUMP[K,N]+(A*EDASH[AA+A]/NOD[M,N]+B*EDASH[AA+B]
/NOD[M+1,N])'ELSE' DELTP[K,N]:=PUMP[K,N]+A*EDASH[AA+A]/
NOD[M,N];
K:=K+1;
'END';
AA:=AA+ND;
```

```
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
REQ[M,N]:=SQRT(ABS(DELTP[M,N])*D[M,N]↑3*RHO*12960000.0*
32.2/(4*L[M,N]*MU↑2));
'IF'REQ[M,N]'LE'129.6'THEN'
'BEGIN'
WRITETEXT('('('2C')'LAMINAR%FLOW%IN%PIPE')');
PRINT(M,3,0);SPACE(5);
WRITETEXT('('('2C')'SECTION%NO.')');
PRINT(N,3,0);NEWLINE(1);
RE[M,N]:=REQ[M,N]↑2/8;
'END''ELSE'
'BEGIN'
RE[M,N]:=-2.5*REQ[M,N]*LN(REL[M,N]/(D[M,N]*3.7)+
1/(1.13*REQ[M,N]));
'IF'RE[M,N]'LE'3000'THEN'
'BEGIN'
WRITETEXT('('('2C')'TRANSITIONAL%FLOW%IN%PIPE')');
PRINT(M,3,0);SPACE(5);
WRITETEXT('('('2C')'SECTION%NO.')');
PRINT(N,3,1);NEWLINE(1);
'END';
'END';
PHI[M,N]:=(REQ[M,N]/RE[M,N])↑2;
'END';
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
V2[M,N]:=SQRT(ABS(DELTP[M,N]*1811.25*D[M,N]↑5*PI↑2/
(PHI[M,N]*RHO*L[M,N])));
V1[M,N]:=V2[M,N];
'END';
'END'FINDFLOW;

'PROCEDURE'FORMADMIT;
'BEGIN''INTEGER'NL,NU;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
Y[M,N]:=1811.25*D[M,N]↑5*PI↑2/(PHI[M,N]*RHO*L[M,N]*V1[M,N]);
```

```
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
OMEGA[M,K]:=0;
K:=1;
'FOR'M:=1'STEP'2'UNTIL'2*BR'DO'
'BEGIN'
A:=ABS(NOD[M,N]);B:=ABS(NOD[M+1,N]);
'IF'B'GT'ND'THEN''GOTO'L6;
OMEGA[A,B]:=OMEGA[B,A]:=-Y[K,N];
OMEGA[B,B]:=OMEGA[B,B]+Y[K,N];
L6: OMEGA[A,A]:=OMEGA[A,A]+Y[K,N];
K:=K+1;
'END';
NL:=1;NU:=ND;
MYTRIX(OMEGA,NU);
PUTARRAY(1,DD,OMEGA);
'END'FORMADMIT;

'PROCEDURE'FORMEDASH;
'BEGIN''REAL''ARRAY'A[1:BR,1:ND,1:1],DUMMY[1:ND,1:1];
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
A[M,K,N]:=0;
P:=1;
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'
'FOR'K:=P'STEP'1'UNTIL'P+1'DO'
'BEGIN'
B:=ABS(NOD[K,N]);
'IF'B'GT'ND'THEN''GOTO'L2;
A[M,B,N]:=NOD[K,N]/B;
'END';
L2:P:=P+2;
'END';
'FOR'M:=1'STEP'1'UNTIL'BR'DO'
Y[M,N]:=Y[M,N]*PUMP[M,N];
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
DUMMY[M,N]:=0.0;
```

```
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
'FOR'K:=1'STEP'1'UNTIL'BR'DO'
DUMMY[M,N]:=A[K,M,N]*Y[K,N]+DUMMY[M,N];
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
FLONOD[M,N]:=FLONOD[M,N]-DUMMY[M,N];
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
DUMMY[M,N]:=0.0;
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
DUMMY[M,N]:=OMEGA[M,K]*FLONOD[K,N]+DUMMY[M,N];
MM:=1;
K:=ND-1;
'FOR'M:=CCC'STEP'1'UNTIL'K+CCC'DO'
'BEGIN'EDASHA[M]:=DUMMY[MM,N];
MM:=MM+1;
'END';
CCC:=CCC+ND;
'END'FORMEDASH;

'PROCEDURE'YBDASH;
'BEGIN''REAL''ARRAY'AC,DUM[1:TOTENODE,1:CUTBR],TAC[1:CUTBR,
                        1:TOTENODE];
'FOR'M:=1'STEP'1'UNTIL'TOTENODE'DO'
'FOR'K:=1'STEP'1'UNTIL'CUTBR'DO'
TAC[K,M]:=AC[M,K]:=DUM[M,K]:=0.0;
A:=1;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
'FOR'K:=A'STEP'1'UNTIL'A+1'DO'
'BEGIN'
B:=ABS(CUTCON[K]);
AC[B,M]:=TAC[M,B]:=CUTCON[K]/B;
'END';
A:=A+2;
'END';
'FOR'B:=1'STEP'1'UNTIL'CUTBR'DO'
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
```

```
'BEGIN'
A:=CC;
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'DUM[M,B]:=OMEGA[M,K]*AC[A,B]+DUM[M,B];
A:=A+1;
'END';
'END';
CC:=CC+ND;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'FOR'B:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
A:=QQ;
'FOR'K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'YB[M,B]:=TAC[M,A]*DUM[K,B]+YB[M,B];
A:=A+1;
'END';
'END';
QQ:=QQ+ND;
'END'YBDASH;

'IF'COUNT=1'THEN'
PIPEDATA;
'IF'DECIDE=2'THEN'
'BEGIN'FINDFLOW;RESULT; 'GOTO'L80;
'END';
'IF'COUNT' GT'1'THEN'
FINDFLOW;FORMADMIT;FORMEDASH;YBDASH;
'END';
'END'CALCULATE;

CALCULATE;
L80:
'END';
'END';
'END';
'BEGIN'
'PROCEDURE'FINDPRESS;
'BEGIN''REAL''ARRAY'ECUTA,ECUT2[1:CUTBR],IDASH,PRESS[1:
                    TOTENODE];
  K:=1;
```

```
'FOR'N:=1'STEP'2'UNTIL'2*CUTBR'DO'
'BEGIN'
A:=ABS(CUTCON[N]);
B:=ABS(CUTCON[N+1]);
ECUTA[K]:=-(EDASHA[A]*A/CUTCON[N]+EDASHA[B]*B/CUTCON[N+1]);
K:=K+1;
'END';
'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
ECUT2[N]:=ECUTA[N]+PUMPC[N];
'FOR'N:=1'STEP'1'UNTIL'TOTENODE'DO'
IDASH[N]:=0.0;
'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
PRESS[N]:=0.0;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
PRESS[M]:=YB[M,N]*ECUT2[N]+PRESS[M];
M:=1;
'FOR'N:=1'STEP'2'UNTIL'2*CUTBR'DO'
'BEGIN'
A:=ABS(CUTCON[N]);B:=ABS(CUTCON[N+1]);
IDASH[A]:=A*PRESS[M]/CUTCON[N]+IDASH[A];
IDASH[B]:=B*PRESS[M]/CUTCON[N+1]+IDASH[B];
M:=M+1;
'END';
'FOR'M:=1'STEP'1'UNTIL'TOTENODE'DO'
EDASH[M]:=0.0;
B:=CC:=C:=1;
'FOR'N:=1'STEP'1'UNTIL'CUTSEC'DO'
'BEGIN''REAL''ARRAY'OMEGA[1:MND[N],1:MND[N]];
GETARRAY(1,B,OMEGA);
ND:=MND[N];MM:=CC;
L30:'FOR'K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
'IF'ND=1'THEN'
'BEGIN'
EDASH[C]:=SINGLE*IDASH[MM];
'GOTO' L40;
'END';
```

```
'FOR'M:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
EDASH[C]:=OMEGA[K,M]*IDASH[MM]+EDASH[C];
MM:=MM+1;
'END';
L40:EDASH[C]:=EDASHA[C]+EDASH[C];
MM:=CC;C:=C+1;
'END';
CC:=CC+ND;
'END';
'END'FINPRESS;

'PROCEDURE'TEST;
'BEGIN'
'INTEGER'SUM2,TRIGGER;
'BEGIN'
'PROCEDURE'CUTFLOW;
'BEGIN''REAL''ARRAY'DUMM[1:CUTBR];
'BEGIN'

'PROCEDURE'CUTPHI;
'BEGIN''REAL''ARRAY'REQC,REC[1:CUTBR];
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
REQC[M]:=SQRT(ABS(DUMM[M])*DC[M]↑3*RHO*12960000.0*32.2/
(4*LC[M]*MU↑2));
'IF'REQC[M]'LE'129.6'THEN'
'BEGIN'
WRITETEXT('('''('2C')'LAMINAR%FLOW%IN%CUTPIPE')');
PRINT(M,3,0);NEWLINE(1);
REC[M]:=REQC[M]↑2/8;
'END''ELSE'
'BEGIN'
REC[M]:=-2.5*REQC[M]*LN(RELC[M]/(DC[M]*3.7)+1/(1.13*REQC
[M]));
'IF'REC[M]'LE'3000'THEN'
'BEGIN'
WRITETEXT('('''('2C')'TRANSITIONAL%FLOW%IN%CUTPIPE')');
PRINT(M,3,0);
```

```
'END';
'END';
PHIC[M]:=(REQC[M]/REC[M])↑2;
'END';
'END'CUTPHI;

M:=1;
'FOR'N:=1'STEP'2'UNTIL'2*CUTBR'DO'
'BEGIN'
A:=ABS(CUTCON[N]);
B:=ABS(CUTCON[N+1]);
DUMM[M]:=A*EDASH[A]/CUTCON[N]+B*EDASH[B]/CUTCON[N+1];
DUMM[M]:=DUMM[M]+PUMPC[M];
M:=M+1;
'END';
CUTPHI;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
VC2[M]:=SQRT(ABS(DUMM[M])*1811.25*DC[M]↑5*PI↑2/(PHIC[M]*
RHO*LC[M]));
VC1[M]:=VC2[M];
'END';
'END';
'END'CUTFLOW;

SUM1:=SUM2:=0.0;
'IF'COUNT'LE'1'THEN'
'FOR'N:=1'STEP'1'UNTIL'TOTENODE'DO'
TEST2[N]:=0;
'FOR'N:=1'STEP'1'UNTIL'TOTENODE'DO'
'BEGIN'
'IF'ABS((ABS(EDASH[N])-TEST2[N])/EDASH[N])'LE'LIMIT'THEN'
SUM2:=SUM2+1;
TEST2[N]:=ABS(EDASH[N]);
'END';
WRITETEXT('('''('1C')'NO.%OF%NODES%CONVERGED')');
PRINT(SUM2,4,0);
```

```
'IF'SUM2'LT'TOTENODE'THEN'
'BEGIN'
COUNT:=COUNT+1;
CUTFLOW;
'END'
'ELSE'DECIDE:=2;
'GOTO'L7;
'END';
'END'TEST;

'PROCEDURE'CUTRESULTS;
'BEGIN'
PAPERTHROW;
WRITETEXT('('('('2C6S')'RESULTS%FOR%CUTPIPES')');
WRITETEXT('('('('2C')'CUTPIPE%NO.%%%%%FLOW%CU.FT./MIN.')');
NEWLINE(2);
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
PRINT(M,3,0);SPACE(3);PRINT(VC2[M],5,3);NEWLINE(1);
'END';
'END'CUTRESULTS;

'IF'DECIDE=2'THEN''GOTO'L70;
'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
ZED[M]:=PHIC[M]*RHO*LC[M]*VC1[M]/(1811.25*DC[M]↑5*PI↑2);
YB[M,M]:=YB[M,M]+ZED[M];
'END';NL:=1;
MYTRIX(YB,CUTBR);
FINDPRESS;
TEST;
L70:CUTRESULTS;
'END';
'END';
'END';
 'END'OF PROGRAM;
```

APPENDIX (12.8)

THE PROGRAM MESHDIAK

```
'BEGIN''COMMENT' A NEW DIAKOPTICS PROGRAM SOLVING BY THE
 MESH METHOD;
      'REAL' RHO, MU, PI, CONVF, GR;
      'INTEGER' CUTSEC,CUTBR,TOTENODE,A,B,N,NN,M,MM,Q,QQ,CC,P,
                COUNT,DECIDE,AA,BB,ND,NU,NS,DD,ZZ,TRIGGER,
                TOTMESH,BR,MESH,F,I,J,K;
      RHO:=READ;
      MU:=READ;
      PI:=READ;
      GR:=32.17;
      CONVF:=READ;
      CUTSEC:=READ;
      CUTBR:=READ;
      TOTENODE:=READ;
      TOTMESH:=READ;
'COMMENT' NOW HAVE TO SET UP DISC ROUTINE;
'BEGIN'
'PROCEDURE'WORKSTORE(N,A,B);
'VALUE'N,B;
'INTEGER'N,B;
'STRING'A;
'EXTERNAL';
'PROCEDURE'PUTARRAY(N,A,B);
'VALUE'N;
'INTEGER'N,A;
'ARRAY'B;
'EXTERNAL';
'PROCEDURE'GETARRAY(N,A,B);
'VALUE'N;
'INTEGER'N,A;
'ARRAY'B;
'EXTERNAL';
'COMMENT' SETUP DISCS;
      WORKSTORE(1,('('ED')'),25000);
      WORKSTORE(2,('('ED')'),25000);
      WORKSTORE(3,('('ED')'),25000);
      WORKSTORE(4,('('ED')'),25000);
```

```
'BEGIN'
'COMMENT'SET UP GLOBAL ARRAYS;
      'REAL''ARRAY'ZHICON[1:CUTBR,1:CUTBR],IDASH[1:TOTMESH],
       CUTMESH[1:TOTMESH,1:CUTBR],LC,DC,VC,PHIC,YC,RELC,
       PUMPC[1:CUTBR],TEST1[1:TOTMESH],MBR,MMESH[1:CUTSEC],
       DETPC[1:CUTBR],MND[1:CUTSEC];
'COMMENT' NOW READ CUTBRANCH CONSTANTS;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'DC[M]:=READ;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'LC[M]:=READ;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'PUMPC[M]:=READ;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'RELC[M]:=READ;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'VC[M]:=1.0;PHIC[M]:=0.002;
'END';
      'FOR'M:=1'STEP'1'UNTIL'TOTMESH'DO'
      'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
      CUTMESH[M,N]:=READ;
'COMMENT' START LOOPING OVER ALL THE SECTIONS,NS IS THE
          COUNTER;
      'FOR'NS:=1'STEP'1'UNTIL'CUTSEC'DO'
'BEGIN'MBR[NS]:=READ;
       MMESH[NS]:=READ;
       MND[NS]:= MBR[NS]-MMESH[NS];
'END';
      COUNT:=0;DECIDE:=0;TRIGGER:=0;
L7:'COMMENT'SET UP COUNTERS;
      CC:=AA:=BB:=ZZ:=DD:=QQ:=F:=1;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
      'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
      ZHICON[M,N]:=0;
      'FOR'NS:=1'STEP'1'UNTIL'CUTSEC'DO'
'BEGIN'
      BR:=MBR[NS];
      MESH:=MMESH[NS];
      ND:= MBR[NS]-MMESH[NS];
'BEGIN'
```

```
'PROCEDURE'CALCULATE;
'BEGIN''REAL''ARRAY'L,D,PHI,V1,V2,REL,DETP,Z[1:BR],
      C[1:ND,1:MESH],SIDASH[1:MESH],DUM[1:ND],OMEGA[1:MESH,
      1:MESH],BT[1:ND,1:ND],NOD[1:2*BR],PUMP[1:BR],IO[1:ND],
      FLONOD[1:ND];
'BEGIN'
'PROCEDURE'SEARCHBT(BT,NOD,ND);
'ARRAY'BT,NOD;
'INTEGER'ND;
'BEGIN''INTEGER''ARRAY'NCP[1:ND+1,1:ND+1],PR[1:ND];
       'INTEGER'I,J,CHECK,S,A,B,K;
      'FOR'I:=1'STEP'1'UNTIL'ND+1'DO'
      'FOR'J:=1'STEP'1'UNTIL'ND+1'DO'
      NCP[I,J]:=0;
      K:=1;
      'FOR'I:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'A:=ABS(NOD[K]);B:=ABS(NOD[K+1]);
      NCP[A,B]:=-I*SIGN(NOD[K]);NCP[B,A]:=-NCP[A,B];
      K:=K+2;
'END';
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'CHECK:=0;J:=ND+1;
      'FOR'K:=1'STEP'1'UNTIL'ND'DO'
      PR[K]:=0;
SS3:  I:=1;
SS1:  A:=ABS(NCP[I,J]);
      'IF'A=0'THEN'
'BEGIN'I:=I+1;'IF'I=ND+2'THEN''GOTO'SS2'ELSE''GOTO'SS1;
'END' 'ELSE''IF'(CHECK-A)#0'THEN'
'BEGIN'PR[A]:=A/(NCP[I,J]);
      'IF'I>J'THEN'PR[A]:=-PR[A];
      J:=I;CHECK:=A;'GOTO'SS3;
'END';S:=I;
SS5:  'IF'I=ND+1'THEN'
'BEGIN'NCP[J,S]:=NCP[S,J]:=0;'GOTO'SS4;
'END';
SS6:  I:=I+1;
      'IF'NCP[I,J]#0'THEN''GOTO'SS1;
      'IF'I=ND+1'THEN''GOTO'SS5'ELSE''GOTO'SS6;
```

```
SS4:  'FOR'K:=1'STEP'1'UNTIL'ND'DO'
      BT[K,J]:=PR[K];
'END';
SS2:  'END'SEARCHBT;

'PROCEDURE'MAKEC(BT,NOD,C,ND);
'ARRAY'BT,NOD,C;
'INTEGER'ND;
'BEGIN''INTEGER'N,M,K,A,B;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
      'FOR'M:=1'STEP'1'UNTIL'ND'DO'
      C[M,N]:=0;
      M:=2*ND+1;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'A:=ABS(NOD[M]);B:=ABS(NOD[M+1]);
      'FOR'K:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'
      'IF'B>ND'THEN''GOTO'SS11;
      C[K,N]:=C[K,N]+SIGN(NOD[M+1])*BT[K,B];
SS11: C[K,N]:=C[K,N]+SIGN(NOD[M])*BT[K,A];
      C[K,N]:=-C[K,N];
'END';M:=M+2;
'END';
'END'MAKEC;

      'IF'COUNT<1'THEN'
'BEGIN''FOR'M:=1'STEP'1'UNTIL'ND' DO'FLONOD[M]:=READ;
       'FOR'M:=1'STEP'1'UNTIL'BR'DO' D[M]:=READ;
       'FOR'M:=1'STEP'1'UNTIL'BR'DO' L[M]:=READ;
       'FOR'M:=1'STEP'1'UNTIL'BR'DO' PUMP[M]:=READ;
       'FOR'M:=1'STEP'1'UNTIL'BR'DO' REL[M]:=READ;
       'FOR'M:=1'STEP'1'UNTIL'2*BR'DO'NOD[M]:=READ;
       'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'V2[M]:=1;PHI[M]:=0.002;
'END';
      SEARCHBT(BT,NOD,ND);
      MAKEC(BT,NOD,C,ND);
```

```
        'FOR'M:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'IO[M]:=0;
        'FOR'N:=1'STEP'1'UNTIL'ND'DO'
        'IF'BT[M,N]#0'THEN'IO[M]:=IO[M]+BT[M,N]*FLONOD[N];
'END';
        PUTARRAY(2,ZZ,D);
        PUTARRAY(2,ZZ,L);
        PUTARRAY(2,ZZ,PUMP);
        PUTARRAY(2,ZZ,REL);
        PUTARRAY(2,ZZ,FLONOD);
        PUTARRAY(2,ZZ,NOD);
        PUTARRAY(2,ZZ,IO);
        PUTARRAY(2,ZZ,C);
'END''ELSE'
'BEGIN'
        GETARRAY(2,ZZ,D);
        GETARRAY(2,ZZ,L);
        GETARRAY(2,ZZ,PUMP);
        GETARRAY(2,ZZ,REL);
        GETARRAY(2,ZZ,FLONOD);
        GETARRAY(2,ZZ,NOD);
        GETARRAY(2,ZZ,IO);
        GETARRAY(2,ZZ,C);
'END';
'BEGIN'
'PROCEDURE'FINDPHI;
'BEGIN''REAL''ARRAY'REQ,RE[1:BR];
        'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN'REQ[M]:=SQRT(ABS(DETP[M])*D[M]↑3*RHO*12960000.0*32.2/
        (4*L[M]*MU↑2));
       'IF'REQ[M]'LE'129.6'THEN'
'BEGIN'WRITETEXT('('''('2C')'LAMINAR%FLOW%IN%PIPE')');
        PRINT(M,3,0);SPACE(5);
        WRITETEXT('('''('2C')'SECTION%NO.')');PRINT(NS,3,0);
        NEWLINE(1);
        RE[M]:=(REQ[M]↑2)/8;
'END' 'ELSE'
'BEGIN'RE[M]:=-2.5*REQ[M]*LN(REL[M]/(D[M]*3.7)+1/(1.13*REQ[
        M]));
```

```
      'IF'RE[M]'LE'3000'THEN'
'BEGIN'WRITETEXT('('''('2C')'TRANSITIONAL%FLOW%IN%PIPE')');
       PRINT(M,3,0);SPACE(5);
       WRITETEXT('('''('2C')'SECTION%NO.')');PRINT(NS,3,0);
       NEWLINE(1);
'END';
'END';PHI[M]:=(REQ[M]/RE[M])↑2;
      V2[M]:=RE[M]*PI*D[M]*MU/(240*RHO);
'END';
'END' FINDPHI;

'PROCEDURE'MESHFLOW;
'COMMENT'THIS PROCEDURE FINDS THE FLOW IN EVERY BRANCH OF A
         SUBDIVISION AFTER THE FIRST PASS FROM IDASH;
'BEGIN'
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
'BEGIN'V2[N]:=0;
      A:=BB;
      'FOR'M:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
'COMMENT' COUNTER BB MUST BE SET OUTSIDE OF CALCULATE;
      'IF'C[N,M]#0'THEN' V2[N]:=V2[N]+IDASH[A]*C[N,M];
      A:=A+1;
'END'; V2[N]:=V2[N]+IO[N];
'END';A:=BB;
      'FOR'M:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'V2[M+ND]:=IDASH[A];
      A:=A+1;
'END'; BB:=BB+MESH;
'BEGIN'

'PROCEDURE'RESULTS;
'BEGIN'WRITETEXT('('''('2C')'RESULTS%FOR%SECTION%')');
       PRINT(NS,3,0);
      ZZ:=ZZ+BR;
       WRITETEXT('('''('2C')'PIPE%NO.%FLOW')');NEWLINE(1);
       'FOR'M:=1'STEP'1'UNTIL'BR'DO'
'BEGIN' PRINT(M,3,0);SPACE(8);PRINT(V2[M],4,3); NEWLINE(1);
'END'; 'GOTO' L100;
'END' RESULTS;
```

```
'COMMENT' MUST FIND PRESSURE IN PIPES FOR FINDPHI;
      GETARRAY(2,ZZ,Z);
       ZZ:=ZZ-BR;
      'FOR'M:=1'STEP'1'UNTIL'BR'DO'
      DETP[M]:=Z[M]*V2[M];
      'IF'TRIGGER=1'THEN'RESULTS;
'END';
'END'MESHFLOW;

'PROCEDURE'FORMCZC;
'BEGIN''FOR'M:=1'STEP'1'UNTIL'BR'DO'
      Z[M]:=ABS(PHI[M]*RHO*L[M]*V2[M]/(1811.25*D[M]↑5*PI↑2));
      PUTARRAY(2,ZZ,Z);
      'FOR' M:=BR-CUTBR+1'STEP'1'UNTIL'BR'DO'
       Z[M]:=0;
'BEGIN' 'INTEGER' K;

'PROCEDURE'MYTRIX(OMEGA,NU);
'VALUE'NU;
'INTEGER'NU;
'ARRAY'OMEGA;
'BEGIN''REAL''ARRAY'V[1:NU-1];
      'REAL'YY,PIVOT;
      'INTEGER'I,J,K;
      'FOR'K:=0'STEP'1'UNTIL'NU-1'DO'
'BEGIN'PIVOT:=1.0/OMEGA[1,1];
      'FOR'I:=2'STEP'1'UNTIL'NU'DO'
      V[I-1]:=OMEGA[1,I];
      'FOR'I:=1'STEP'1'UNTIL'NU-1'DO'
'BEGIN'OMEGA[I,NU]:=YY:=-V[I]*PIVOT;
      'FOR'J:=I'STEP'1'UNTIL'NU-1'DO'
      OMEGA[I,J]:=OMEGA[I+1,J+1]+V[J]*YY;
'END';OMEGA[NU,NU]:=-PIVOT;
'END';'FOR'I:=1'STEP'1'UNTIL'NU'DO'
      'FOR'J:=I'STEP'1'UNTIL'NU'DO'
      OMEGA[I,J]:=OMEGA[J,I]:=-OMEGA[I,J];
'END' MYTRIX;
```

```
      'FOR'M:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN''FOR'K:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'OMEGA[M,K]:=0;
      'FOR'N:=1'STEP'1'UNTIL'ND'DO'
       OMEGA[M,K]:=OMEGA[M,K]+C[N,M]*Z[N]*C[N,K];
'END';OMEGA[M,M]:=OMEGA[M,M]+Z[ND+M];
'END';
      MYTRIX(OMEGA,MESH);
      PUTARRAY(1,DD,OMEGA);
'COMMENT'THIS STORES YALPHA FOR INTERCONNECTION STEP;
'COMMENT'NOW FORM (E-ZI) FOR ALL THE BRANCHES IN THE SUBNTWK;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'DUM[N]:=0;
      'FOR'M:=1'STEP'1'UNTIL'ND'DO'
      DUM[N]:=DUM[N]+C[M,N]*(PUMP[M]-Z[M]*IO[M]);
      DUM[N]:=DUM[N]+PUMP[ND+N]
'END';'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'SIDASH[N]:=0;
      'FOR'M:=1'STEP'1'UNTIL'MESH'DO'
      SIDASH[N]:=SIDASH[N]+OMEGA[N,M]*DUM[M];
'END';
'COMMENT'PUT INTO IDASH FOR OUTSIDE ROUTINE USING COUNTER CC;
      M:=1;
      N:=MESH-1;
      'FOR'I:=CC'STEP'1'UNTIL'N+CC'DO'
'BEGIN'IDASH[I]:=SIDASH[M];M:=M+1;
'END'; CC:=CC+MESH;
'END';
'END'FORMCZC;

'PROCEDURE' HICON;
'BEGIN'
'COMMENT' THIS PROCEDURE FINDS ALL THE SUBDIVISIONS AND GIVES
          THE OVERALL CONNECTION MATRIX FOR HILEVCON;
'COMMENT' ZHICON MUST HAVE BEEN ZEROED OUTSIDE OF CALCULATE;
      'REAL''ARRAY'DUM5[1:MESH,1:CUTBR];
      'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'A:=QQ;DUM5[K,M]:=0;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
```

```
'BEGIN'DUM5[K,M]:=DUM5[K,M]+OMEGA[K,N]*CUTMESH[A,M];
       A:=A+1;
'END';
'END';
      'FOR'K:=1'STEP'1'UNTIL'CUTBR'DO'
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'A:=QQ;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'ZHICON[K,M]:=ZHICON[K,M]+CUTMESH[A,K]*DUM5[N,M];
       A:=A+1;
'END';
'END';
      QQ:=QQ+MESH;
'END'  HICON;

      'IF'COUNT#0'THEN'
'BEGIN'MESHFLOW;
       FINDPHI;
'END'; FORMCZC;
        HICON;
'END';
'END';
L100:'END' CALCULATE;

       CALCULATE;
'END';
'END';
'BEGIN'
'PROCEDURE'INTERCONECT;
'COMMENT' SET UP ISTAR  BY ALPH*IDASH INVERT ZHICON AND
          MULTIPLY;
'BEGIN''REAL''ARRAY'IDASH2[1:TOTMESH],EDASH[1:CUTBR],
      EDASH2[1:TOTMESH],IDASH1[1:TOTMESH];
'BEGIN'

'PROCEDURE'MYTRIZ(ZHICON,NU);
'VALUE'NU;
'INTEGER'NU;
'ARRAY'ZHICON;
'BEGIN''REAL''ARRAY'V[1:NU-1];
      'REAL'YY,PIVOT;
      'INTEGER'I,J,K;
```

```
      'FOR'K:=0'STEP'1'UNTIL'NU-1'DO'
'BEGIN'PIVOT:=1.0/ZHICON[1,1];
      'FOR'I:=2'STEP'1'UNTIL'NU'DO'
      V[I-1]:=ZHICON[1,I];
      'FOR'I:=1'STEP'1'UNTIL'NU-1'DO'
'BEGIN'ZHICON[I,NU]:=YY:=-V[I]*PIVOT;
      'FOR'J:=I'STEP'1'UNTIL'NU-1'DO'
      ZHICON[I,J]:=ZHICON[I+1,J+1]+V[J]*YY;
'END';ZHICON[NU,NU]:=-PIVOT;
'END';'FOR'I:=1'STEP'1'UNTIL'NU'DO'
      'FOR'J:=I'STEP'1'UNTIL'NU'DO'
      ZHICON[I,J]:=ZHICON[J,I]:=-ZHICON[I,J];
'END' MYTRIZ;

      'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'IDASH1[N]:=0;
      'FOR'M:=1'STEP'1'UNTIL'TOTMESH'DO'
      IDASH1[N]:=IDASH1[N]-CUTMESH[M,N]*IDASH[M];
'END';'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
       ZHICON[N,N]:=ZHICON[N,N]+YC[N];
      MYTRIZ(ZHICON,CUTBR);
      'FOR'N:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'EDASH[N]:=0;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
      EDASH[N]:=EDASH[N]+ZHICON[N,M]*IDASH1[M];
'END';
      'FOR'N:=1'STEP'1'UNTIL'TOTMESH'DO'
'BEGIN'EDASH2[N]:=0;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
      EDASH2[N]:=EDASH2[N]+CUTMESH[N,M]*EDASH[M];
'END';
'COMMENT'EDASH2 IS THE CURRENT COMPONENT DUE TO INTERCONNECTION;
'COMMENT'NOW BRING DOWN THE STORE SOLUTION EQUATIONS;
      'FOR'M:=1'STEP'1'UNTIL'TOTMESH'DO'
      IDASH2[M]:=0;
      DD:=1;CC:=1;
      'FOR'NS:=1'STEP'1'UNTIL'CUTSEC'DO'
'BEGIN'      'REAL''ARRAY'OMEGA[1:MMESH[NS],1:MMESH[NS]];
      GETARRAY(1,DD,OMEGA);
```

```
      MESH:=MMESH[NS];
       J:=CC;
      'FOR'N:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
      'FOR'K:=1'STEP'1'UNTIL'MESH'DO'
'BEGIN'
      IDASH2[J]:=OMEGA[N,K]*EDASH2[CC-1+K]+IDASH2[J];
'END';
      IDASH[J]:=IDASH[J]+IDASH2[J];
      J:=J+1;
'END';
        CC:=CC+MESH;
'END';
'END';
'END'INTERCONECT;

'PROCEDURE'CUTVALUES;
'BEGIN'
'PROCEDURE'FINDCUTPHI;
'BEGIN''REAL''ARRAY'REQC,REC[1:CUTBR];
      'FOR' M:=1'STEP'1'UNTIL' CUTBR'DO'
'BEGIN'REQC[M]:=SQRT(ABS(DETPC[M])*DC[M]↑3*RHO*12960000.0*
       32.2/(4*LC[M]*MU↑2));
      'IF'REQC[M]'LE'129.6'THEN'
'BEGIN'WRITETEXT('('''('2C')'LAMINAR%FLOW%IN%CUTPIPE')');
       PRINT(M,3,0);NEWLINE(1);
      REC[M]:=(REQC[M]↑2)/8;
'END' 'ELSE'
'BEGIN'REC[M]:=-2.5*REQC[M]*LN(RELC[M]/(DC[M]*3.7)+
       1/(1.13*REQC[M]));
      'IF'REC[M]'LE'3000'THEN'
'BEGIN'WRITETEXT('('''('2C')'TRANSITIONAL%FLOW%IN%CUTPIPE')');
       PRINT(M,3,0);NEWLINE(1);
'END';
'END'; PHIC[M]:=(REQC[M]/REC[M])↑2;
      VC[M]:=REC[M]*PI*DC[M]*MU/(240*RHO);
'END';
'END' FINDCUTPHI;
```

```
     'FOR' N:=1'STEP'1'UNTIL' CUTBR'DO'
'BEGIN' VC[N]:=0;
      'FOR' M:=1'STEP'1'UNTIL' TOTMESH'DO'
      VC[N]:=VC[N]+CUTMESH[M,N]*IDASH[M];
'END';
'COMMENT' KNOWING FLOW IN CUTPIPES FROM CALCULATE, FIND
          PRESSURE DROP HENCE NEW YC;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
      DETPC[M]:=VC[M]/YC[M];
      FINDCUTPHI;
'BEGIN'
'PROCEDURE'CUTRESULTS;
'BEGIN'WRITETEXT('('''('2C')'RESULTS%FOR'CUTBRANCHES')');
       WRITETEXT('('''('2C')'CUTBRANCH%NO.%%%%%%%%%FLOW%%%
       %%PRESSURE')');
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
'BEGIN'
       NEWLINE(1);
      PRINT(VC[M],5,4);SPACE(8);PRINT(DETPC[M],5,4);
'END';
      'GOTO' L200;
'END' CUTRESULTS;

      'IF'TRIGGER=1'THEN'CUTRESULTS;
'END';
'END' CUTVALUES;

'PROCEDURE'TESTS;
'BEGIN''INTEGER'SUM1;
      WRITETEXT('('''('2C')'TEST1')');
      'IF'COUNT'LT'1'THEN''FOR'M:=1'STEP'1'UNTIL'TOTMESH'DO'
      TEST1[M]:=1;
      SUM1:=0;
      'FOR'M:=1'STEP'1'UNTIL'TOTMESH'DO'
'BEGIN''IF'ABS(ABS(IDASH[M])-TEST1[M])/TEST1[M]'LE'CONVF'THEN'
      SUM1:=SUM1+1;
      TEST1[M]:=ABS(IDASH[M]);
'END';'IF'SUM1'LT'TOTMESH'THEN'
'BEGIN'DECIDE:=1;TRIGGER:=0;
      COUNT:=COUNT+1;
      'GOTO'L7;
```

```
'END';
      TRIGGER:=1;
      'GOTO'L7;
'END' TESTS;
      'FOR'M:=1'STEP'1'UNTIL'CUTBR'DO'
      YC[M]:=1811.25*DC[M]↑5*PI↑2/(PHIC[M]*RHO*LC[M]*
      ABS(VC[M]));
      'IF' TRIGGER=1'THEN''GOTO' L80;
      INTERCONECT;
L80:  CUTVALUES;
      TESTS;
'END';
L200:'END';
'END';
'END';
```

APPENDIX (13)

TABLES OF RESULTS

Table (1) The connection list for the three cases of overlap for the network TEST1

Branch Number	Overlap Choice					
	Minimum		Arbitrary		Maximum	
1	6	-22	-9	10	-1	2
2	6	-9	-1	2	-2	5
3	-9	10	-2	3	3	-5
4	-8	9	-3	4	-3	4
5	-10	12	-10	12	-4	18
6	-6	11	-17	19	17	-18
7	-4	19	-4	19	-17	19
8	-18	19	-18	19	-19	20
9	17	-18	7	-20	7	-20
10	16	-19	13	-15	7	-21
11	15	-16	8	-22	-16	21
12	-16	20	-16	20	15	-16
13	-16	21	-8	10	14	-15
14	7	-21	7	-21	-13	14
15	-13	14	-13	14	-12	13
16	5	-6	11	-12	11	-12
17	-1	6	-1	6	-10	11
18	-2	5	-2	5	-8	10
19	3	-5	-12	13	-8	9
20	4	-5	-19	20	6	-9
21	14	-16	-13	21	6	-22
22	-1	22	-1	22	-1	22
23	-1	2	6	-22	-1	6
24	-2	3	6	-9	-2	3
25	-3	4	-6	11	4	-5

Table (1) (Continued)

Branch Number	Overlap Choice					
	Minimum		Arbitrary		Maximum	
26	-4	18	5	-6	5	-6
27	-17	19	4	-5	-6	11
28	-19	20	3	-5	-9	10
29	7	-20	-4	18	8	-22
30	14	-15	-11	17	-10	12
31	13	-15	16	-19	-11	17
32	8	-22	-16	21	-18	19
33	-10	11	15	-16	-4	19
34	-8	10	14	-16	16	-19
35	-13	21	14	-15	-16	20
36	-11	17	17	-18	14	-16
37	11	-12	-8	9	13	-15
38	-12	13	-10	11	-13	21

Table (2) The node to datum flow vector for all overlap cases of TEST1 .

Node No	Flow cu ft/min
1	120
2	0
3	0
4	-120
5	0
6	0
7	-240
8	210
9	0
10	0
11	0
12	90
13	0
14	-60
15	0
16	0
17	0
18	0
19	0
20	0
21	0

Table (3) The results obtained by Middleton and Gay (45) for the analysis of the three overlap conditions of TEST1 (program similar to HCMESHIN)

Overlap Condition	Number of Iterations	Time (Minutes)
Minimum	15	30
Arbitrary	35	76
Maximum	50 (not converged)	120 (stopped)

Table (4) The results obtained analysing the three overlap conditions of TEST1 using HCMESHIN

Overlap condition	Number of iterations		Time (minutes)
	Inner Cycle	Outer Cycle	
Minimum	54	5	0.416
Arbitrary	123	5	0.917
Maximum	140	5	1.000

Table (5) The results obtained analysing the three overlap conditions of TEST1 using HCMESHOUT

Overlap condition	Number of iterations	Time (minutes)
Minimum	22	0.256
Arbitrary	65	0.716
Maximum	73	0.833

Table (6) The individual pipe flows for the three cases of overlap of HCMESHOUT compared to the minimum overlap case of Middleton (6)

HCMESHOUT				Middleton (6)	
Minimum Overlap		Arbitrary Overlap		Maximum Overlap	Minimum Overlap
Branch No	Flow cu ft/min	Branch No	Flow cu ft/min	Branch No	Flow cu ft/min
23	88.42	2	88.42	1	88.56
24	50.87	3	50.87	24	50.94
19	31.45	28	31.47	3	31.56
18	37.55	18	37.55	2	37.62
16	84.00	26	83.99	26	83.88
17	45.55	17	45.58	23	45.78
22	13.97	22	14.00	22	14.40
1	48.52	23	48.49	21	48.30
32	62.50	11	62.49	29	62.70
4	69.52	37	69.52	19	69.48
2	38.68	24	38.66	20	38.58
6	48.75	25	48.73	27	48.78
3	30.84	1	30.85	28	30.90
34	77.98	13	77.98	18	77.82
33	54.44	38	54.46	17	53.88
37	1.74	16	1.75	16	2.874
5	54.38	5	54.39	30	54.84
26	5.51	29	5.51	5	5.42
7	46.92	7	46.91	33	47.04
8	46.34	8	46.34	32	46.52
9	40.83	36	40.83	6	41.09
27	64.11	6	64.10	7	64.44

Table (6) (Continued)

HCMESHOUT						Middleton (6)
Minimum Overlap		Arbitrary Overlap		Maximum Overlap		Minimum Overlap
Branch No	Flow cu ft/min	Branch No	Flow cu ft/min	Branch No	Flow cu ft/min	Flow cu ft/min
28	84.56	20	84.56	8	84.55	84.84
10	72.80	31	72.80	34	72.80	73.20
12	38.14	12	38.13	35	38.14	39.93
29	122.69	9	122.70	9	122.69	122.78
14	117.30	14	117.31	10	117.30	117.2
13	50.35	32	50.35	11	50.34	50.60
21	1.81	34	1.81	36	1.80	2.06
11	17.48	33	17.50	12	17.49	17.40
30	17.66	35	17.66	13	17.67	17.56
35	66.95	21	66.96	38	66.96	66.60
15	40.53	15	40.53	14	40.53	40.38
31	35.15	10	35.16	37	35.16	34.96
38	142.63	19	142.64	15	142.65	142.00
25	82.35	4	82.35	4	82.34	82.50
20	90.08	27	90.07	25	90.07	90.00
36	104.93	30	104.94	31	104.95	105.5

Table (7) The individual pipe flows for the three cases of overlap of HCMESHIN for comparison to HCMESHOUT and the minimum overlap case of Middleton in table (6)

Minimum Overlap		Arbitrary Overlap		Maximum Overlap	
Branch No	Flow cu ft/min	Branch No	Flow cu ft/min	Branch No	Flow cu ft/min
23	88.42	2	88.43	1	88.46
24	50.88	3	50.89	24	50.89
19	31.48	28	31.48	3	31.48
18	37.55	18	37.55	2	37.57
16	84.01	26	84.01	26	83.99
17	45.56	17	45.56	23	45.60
22	13.98	22	13.98	22	14.06
1	48.52	23	48.52	21	48.47
32	62.50	11	62.51	29	62.53
4	69.52	37	69.52	19	69.51
2	38.68	24	38.68	20	38.67
6	48.75	25	48.75	27	48.75
3	30.84	1	30.84	28	30.84
34	77.98	13	77.97	18	77.97
33	54.44	38	54.44	17	54.44
37	1.74	16	1.74	16	1.74
5	54.37	5	54.37	30	54.37
26	5.51	29	5.52	5	5.52
7	46.92	7	46.92	33	46.93
8	46.35	8	46.35	32	46.35
9	40.83	36	40.83	6	40.83
27	64.10	6	64.10	7	64.10
28	84.56	20	84.56	8	84.56

Table (7) (Continued)

Minimum Overlap		Arbitrary Overlap		Maximum Overlap	
Branch No	Flow cu ft/min	Branch No	Flow cu ft/min	Branch No	Flow cu ft/min
10	72.81	31	72.81	34	72.81
12	38.14	12	38.14	35	38.14
29	122.70	9	122.69	9	122.70
14	117.30	14	117.31	10	117.30
13	50.35	32	50.35	11	50.35
21	1.81	34	1.81	36	1.81
11	17.49	33	17.49	12	17.48
30	17.67	35	17.67	13	17.67
35	66.96	21	66.95	38	66.96
15	40.53	15	40.53	14	40.52
31	35.15	10	35.15	37	35.15
38	142.63	19	142.63	15	142.63
25	82.36	4	82.36	4	82.37
20	90.08	27	90.08	25	90.08
36	104.93	30	104.93	31	104.93

Table (8) The percentage reduction in convergence time in terms of HCMESHIN for the improved program HCMESHOUT

Overlap condition	Percentage reduction in convergence time for HCMESHOUT
Minimum	38.5
Arbitrary	21.9
Maximum	16.6

Table (9) The connection list for the three cases of overlap for the network due to Knights and Allen

Branch Number	Minimum		Arbitrary		Maximum	
1	-6	20	4	-20	4	-20
2	-4	20	-4	5	-4	5
3	-7	20	1	-5	1	-5
4	-16	20	-1	2	-1	2
5	-11	20	-2	3	-2	3
6	-9	11	-3	18	-3	18
7	9	-10	6	-20	-18	19
8	8	-9	7	-20	16	-19
9	-12	14	7	-8	-16	17
10	-13	14	-8	9	15	-17
11	-14	15	16	-20	14	-15
12	-14	16	-16	19	13	-14
13	-16	17	-16	17	12	-13
14	-16	19	15	-17	11	-12
15	-16	18	-14	15	9	-11
16	3	-4	-12	14	-9	10
17	2	-3	-11	12	8	-10
18	-4	5	-12	13	7	-8
19	1	-5	10	-13	6	-7
20	-1	2	5	-6	5	-6
21	-6	7	3	-4	-6	20
22	-7	8	-18	19	-7	20
23	-8	10	16	-18	-8	9
24	-10	13	-6	7	16	-20
25	-9	13	11	-20	11	-20

Table (9) (Continued)

Branch Number	Minimum		Arbitrary		Maximum	
26	-11	12	-8	10	-9	13
27	-12	13	-9	10	-10	13
28	-15	17	-9	13	-12	14
29	-17	19	-13	14	-14	16
30	-18	19	-14	16	-16	18
31	-3	18	-17	19	3	-4
32	5	-6	-9	11	-17	19

Table (10) The dimensions of the network due to Knights and Allen for the case of minimum overlap

Branch Number	Length ft	Diameter ft
1	31 680	0.8541
2	21 120	1.0208
3	42 240	0.8541
4	42 240	1.2812
5	29 040	1.0208
6	10 560	1.0208
7	10 560	1.0208
8	5 280	1.0208
9	5 280	0.5104
10	15 840	0.6671
11	5 280	0.5104
12	10 560	1.0208
13	26 400	1.0208
14	36 960	0.6771
15	21 120	1.0208
16	5 280	0.6711
17	16 840	0.6711
18	10 560	0.6711
19	5 280	0.5104
20	10 560	0.3437
21	26 400	0.5104
22	10 560	0.3437
23	31 680	0.5104
24	26 400	0.3437
25	31 680	1.2813

Table (10) (Continued)

Branch Number	Length ft	Diameter ft
26	10 560	0.6671
27	21 120	0.6771
28	21 120	0.8541
29	47 520	0.5104
30	7 920	0.3437
31	21 120	1.0208
32	10 560	0.6771

Table (11) The node to datum flows for the network due to Knights and Allen (35)

Node No	Flow cu ft/min
1	-166.67
2	-250.0
3	0
4	-250.0
5	0
6	-500.0
7	-333.3
8	0
9	3166.0
10	-500.0
11	-500.0
12	0
13	-1666.7
14	-833.4
15	-333.3
16	3333.0
17	-666.7
18	0
19	-500.0

Table (12) The results for the maximum overlap choice of the Knights and Allen network as a water distribution network. Branch flows in cu ft/min

Branch No	NODEFLAN	MESHFLAN	LATTNODEFLAN	HCMESHOUT
1	-383.53	-383.63	-383.53	-383.54
2	-255.46	-255.57	-255.45	-255.60
3	-151.60	-151.68	-151.60	-151.661
4	15.02	15.02	15.02	15.04
5	264.92	265.06	264.92	265.04
6	386.68	386.85	386.68	387.06
7	-109.09	-109.16	-109.09	-108.98
8	314.17	314.35	314.17	313.93
9	-855.47	-856.14	-855.47	-850.83
10	-112.70	-112.68	-112.70	-106.74
11	220.47	220.69	220.47	226.26
12	-55.04	-55.08	-55.04	-53.68
13	118.91	118.99	118.91	118.95
14	230.88	230.94	230.88	229.45
15	1080.38	1081.25	1080.38	1079.74
16	-502.49	-502.83	-502.49	-502.80
17	42.50	42.53	42.50	42.54
18	-92.20	-92.23	-92.20	-92.18
19	-37.10	-37.12	-37.10	-37.20
20	103.88	103.91	103.88	103.90
21	358.81	358.95	358.81	358.90
22	278.07	278.17	278.07	278.02
23	134.69	134.75	134.69	134.72
24	669.32	669.68	669.32	670.16
25	-349.97	-350.04	-349.97	-350.30

Table (12) (Continued)

Branch No	NODEFLAN	MESHFLAN	LATTNODEFLAN	HCMESHOUT
26	-1446.30	-1447.24	-1446.30	-1448.74
27	-45.25	-45.27	-45.25	-45.34
28	-111.99	112.06	111.99	110.50
29	996.32	996.89	996.32	1002.04
30	-495.72	-495.94	-495.72	-496.04
31	121.81	121.86	121.80	122.02
32	-76.56	-76.58	-76.56	-77.09

Table (13) The results obtained using HCMESHIN for the analysis of the network due to Knights and Allen (35) under three conditions of overlap

Overlap condition	Number of iterations		Time (minutes)
	Inner Cycle	Outer Cycle	
Minimum	30	5	0.35
Arbitrary	133	5	0.884
Maximum	156	5	1.15

Table (14) The results obtained using HCMESHOUT for the analysis of the network due to Knights and Allen (35) under three conditions of overlap

Overlap condition	Number of iterations	Time (minutes)
Minimum	16	0.266
Arbitrary	103	0.684
Maximum	114	1.02

Table (15) The percentage reduction in solution time for the program HCMESHOUT relative to HCMESHIN for the network due to Knights and Allen (35)

Overlap condition	Percentage reduction in solution time for HCMESHOUT
Minimum	24.0
Arbitrary	22.6
Maximum	11.2

Table (16) The connection lists for the three choices of overlap for the network W S Atkins No 1

Branch Number	Minimum Overlap		Arbitrary Overlap		Maximum Overlap	
1	-1	2	-1	2	-1	2
2	-2	3	-2	3	-2	3
3	-3	5	-20	21	-3	5
4	-5	6	-5	6	-6	7
5	-6	22	-6	22	-6	22
6	-21	22	-21	22	-21	22
7	-3	4	-3	4	-3	4
8	-4	8	-4	8	-4	8
9	-7	8	-7	8	-7	8
10	-4	9	-7	9	-20	21
11	-9	23	-9	23	-9	23
12	-9	10	-9	10	-9	10
13	-10	11	-10	11	-10	11
14	-11	12	-11	12	-11	12
15	-12	17	-12	17	-17	19
16	-12	16	-17	19	-16	17
17	-12	13	-16	17	-12	13
18	-13	15	-13	15	-13	15
19	-16	18	-16	18	-16	18
20	-18	19	-15	16	-15	16
21	-19	20	-19	20	-19	20
22	-14	15	-14	15	-14	15
23	-15	16	-12	13	-18	19
24	-16	17	-12	16	-12	17
25	-17	19	-18	19	-12	16

Table (16) (Continued)

Branch Number	Minimum Overlap		Arbitrary Overlap		Maximum Overlap	
26	-20	21	-4	9	-5	6
27	-6	7	-6	7	-4	9
28	-7	9	-3	5	-7	9

Table (17) The dimensions of the network W S Atkins No 1 for the case of maximum overlap

Branch Number	Length ft	Diameter ft	Roughness
1	940	1.0	0.003
2	7 040	1.0	0.003
3	2 110	1.0	0.004
4	880	0.5	0.005
5	2 650	0.833	0.005
6	1 700	0.833	0.005
7	2 700	1.25	0.004
8	500	0.75	0.005
9	2 480	0.5	0.005
10	3 850	0.666	0.004
11	2 020	0.75	0.003
12	1 420	1.25	0.005
13	1 310	1.25	0.005
14	1 720	1.0	0.005
15	1 450	0.75	0.004
16	1 450	0.75	0.004
17	360	0.75	0.02
18	700	0.5	0.02
19	5 100	0.666	0.001
20	2 130	0.75	0.005
21	3 070	0.5	0.005
22	1 440	0.5	0.02
23	3 800	0.5	0.005
24	3 980	0.5	0.005
25	1 360	1.0	0.004
26	2 380	1.0	0.005

Table (17) (Continued)

Branch Number	Length ft	Diameter ft	Roughness
27	1 080	1.25	0.005
28	2 790	0.5	0.02

Table (18) The node to datum flows for the network W S Atkins No 1

Node No	Flow cu ft/min
1	184.85
2	-27.02
3	-11.89
4	-17.29
5	-15.13
6	-28.10
7	-16.21
8	0
9	-19.45
10	-9.73
11	-15.13
12	-11.89
13	0
14	-19.45
15	-12.97
16	-8.65
17	-11.89
18	-64.86
19	-57.29
20	-14.05
21	0
22	-21.62

Table (19) The connection lists for the three cases of overlap for the network due to Dolan

Branch Number	Minimum Overlap		Arbitrary Overlap		Maximum Overlap	
1	-1	2	-1	2	-1	2
2	-2	3	-2	3	-2	3
3	-1	4	-3	4	-3	4
4	-2	29	-4	5	-4	5
5	28	-29	-5	6	-5	6
6	27	-28	-6	7	-6	7
7	26	-27	-7	8	-7	8
8	25	-26	-8	9	-8	9
9	24	-25	-9	10	-9	10
10	19	-26	-10	11	-10	11
11	-19	23	19	-23	-11	12
12	22	-23	-12	13	-12	13
13	-20	21	-13	14	-13	14
14	-19	20	-14	15	-14	15
15	-18	19	-15	16	-15	16
16	-17	18	-16	17	-16	17
17	-15	16	-17	18	-17	18
18	-14	15	-18	19	-18	19
19	-13	14	-19	20	-19	20
20	9	-13	-20	21	-20	21
21	-12	13	-21	22	-21	22
22	-9	10	-9	13	-22	23
23	-10	11	-22	24	-22	24
24	8	-9	-24	25	-24	25
25	7	-8	-25	26	-25	26

Table (19) (Continued)

Branch Number	Minimum Overlap		Arbitrary Overlap		Maximum Overlap	
26	4	-8	-26	27	-26	27
27	-4	5	-27	28	-27	28
28	-5	6	2	-29	-28	29
29	-6	7	-19	26	-19	26
30	3	-4	-22	23	19	-23
31	-3	28	3	-28	3	-28
32	22	-24	-28	29	2	-29
33	21	-22	-1	4	-1	4
34	16	-17	-4	8	-4	8
35	11	-12	-11	12	-9	13

Table (20) The dimensions of the network due to Dolan

Maximum overlap case

Branch Number	Length ft	Diameter ft
1	2 500	0.667
2	1 200	0.667
3	14 100	1.333
4	1 200	1.0
5	2 000	1.0
6	2 000	0.5
7	3 000	1.0
8	4 000	1.0
9	3 300	1.0
10	2 100	1.0
11	3 900	1.0
12	8 800	1.0
13	3 300	1.167
14	1 000	1.333
15	3 000	1.667
16	4 100	1.0
17	5 000	1.0
18	1 500	1.0
19	1 000	0.667
20	5 000	0.833
21	2 500	0.833
22	2 000	0.5
23	1 600	0.667
24	1 500	0.667
25	2 200	0.833

Table (20) (Continued)

Branch Number	Length ft	Diameter ft
26	2 000	1.0
27	2 200	1.0
28	5 300	0.833
29	3 400	1.0
30	1 000	0.5
31	8 000	1.333
32	4 500	0.667
33	2 600	1.333
34	4 500	1.0
35	9 300	1.0

Table (21) The node to datum flows for the network due to Dolan

Node No	Flow cu ft/min.
1	0
2	-12.5
3	0
4	-2.083
5	-6.25
6	0
7	-14.58
8	0
9	0
10	-20.30
11	-93.75
12	-2.083
13	-6.25
14	-2.083
15	-2.083
16	206.25
17	-8.33
18	-6.25
19	0
20	0
21	-12.5
22	0
23	-4.167
24	-2.083
25	-4.167
26	0
27	-2.621
28	-4.167

Table (22) The reduction in solution time for the program HCMESHOUT analysing the three cases of overlap of W G Atkins No 1

Overlap case	Percentage reduction in solution time
Minimum	27.0
Arbitrary	33.3
Maximum	5.5

Table (23) The reduction in solution time for the program HCMESHOUT analysing the three cases of overlap of the network due to Dolan

Overlap case	Percentage reduction in solution time
Minimum	17.5
Arbitrary	5.5
Maximum	31.0

Table (24) The individual pipe flows obtained for NODEFLAN, MESHFLAN and LATTCLAN for the maximum overlap case of TEST1 figure (5.2.4)

Branch No	NODEFLAN		MESHFLAN		LATTNODEFLAN	
	Flow cu ft/min	Pressure lb/ft^2	Flow cu ft/min	Pressure lb/ft^2	Flow cu ft/min	Pressure lb/ft^2
1	-88.40	148.65	-88.42	148.71	-88.40	148.65
2	-37.54	31.65	-37.54	31.66	-37.54	31.65
3	-31.47	23.06	-31.47	23.06	-31.47	23.06
4	-82.34	130.67	-82.35	130.73	-82.33	130.67
5	-5.51	1.04	-5.51	1.04	-5.51	1.04
6	40.82	36.81	40.83	36.82	40.82	36.81
7	-64.09	83.05	-64.10	83.07	-64.09	83.05
8	-84.54	137.09	-84.56	137.14	-84.54	137.09
9	-122.66	269.42	-122.69	269.54	-122.66	269.42
10	-117.28	248.29	-117.31	248.41	-117.28	248.29
11	-50.34	53.69	-50.34	53.70	-50.34	53.69
12	17.48	8.06	17.49	8.05	17.49	8.06
13	-17.66	8.20	-17.67	8.20	-17.67	8.20
14	-40.53	36.33	-40.53	36.33	-40.53	36.33
15	-142.60	354.33	-142.63	354.49	-142.60	354.33
16	-1.74	0.14	-1.74	0.14	-1.74	0.14
17	-54.44	61.83	-54.44	61.85	-54.44	61.83
18	-77.97	118.39	-77.98	118.42	-77.97	118.39
19	-69.50	96.16	-69.52	96.20	-69.51	96.16
20	-38.68	33.40	-38.68	33.41	-38.68	33.40
21	-48.52	50.23	-48.52	50.25	-48.52	50.24
22	13.97	5.40	13.98	5.41	13.98	5.40

Table (24) (Continued)

Branch No	NODEFLAN		MESHFLAN		LATTNODEFLAN	
	Flow cu ft/min	Pressure lb/ft^2	Flow cu ft/min	Pressure lb/ft^2	Flow cu ft/min	Pressure lb/ft^2
23	-45.54	44.83	-45.55	44.84	-45.55	44.83
24	-50.87	54.72	-50.86	54.73	-50.87	54.72
25	-90.06	153.73	-90.08	153.80	-90.06	153.74
26	-83.99	135.47	-84.01	135.53	-83.99	135.47
27	-48.74	50.66	-48.75	50.67	-48.75	50.66
28	-30.84	22.23	-30.84	22.23	-30.84	22.23
29	62.49	79.33	62.50	79.35	62.49	79.33
30	-54.37	61.69	-54.38	61.71	-54.37	61.69
31	-104.91	202.78	-104.93	202.88	-104.91	202.78
32	-46.34	46.24	-46.34	46.25	-46.34	46.23
33	-46.91	47.28	-46.92	47.30	-46.91	47.28
34	-72.79	104.53	-72.80	104.58	-72.79	104.53
35	-38.13	32.56	-38.13	32.56	-38.13	32.56
36	-1.80	0.15	-1.81	0.15	-1.81	0.15
37	35.14	28.12	35.15	28.12	35.15	28.12
38	-66.95	89.86	-66.96	89.87	-66.95	89.86

Table (25) Convergence of the orthogonal matrix programs for the three overlap cases of TEST1

Choice of Overlap	Network Name					
	NODEFLAN		MESHFLAN		LATTNODEFLAN	
	Convergence (mins)	No of iterations	Convergence (mins)	No of iterations	Convergence (mins)	No of iterations
Minimum	0.58	11	1.07	13	2.02	11
Arbitrary	0.63	11	1.0	13	2.28	11
Maximum	0.5	11	1.05	14	2.44	11

Table (26) Convergence of NODEFLAN, MESHFLAN AND LATTNODEFLAN for the three cases of overlap for Knights and Allen

Choice of Overlap	Program					
	NODEFLAN		MESHFLAN		LATTNODEFLAN	
	Con-vergence (mins)	No of itera-tions	Con-vergence (mins)	No of itera-tions	Con-vergence (mins)	No of itera-tions
Minimum	0.468	12	0.666	13	1.58	12
Arbitrary	0.455	12	0.764	14	1.58	12
Maximum	0.45	11	0.767	13	1.60	11

Table (27) Convergence of NODEFLAN, MESHFLAN and LATTNODEFLAN for the three cases of overlap for the network W S Atkins No 1

Choice of Overlap	Program					
	NODEFLAN		MESHFLAN		LATTNODEFLAN	
	Con-vergence (mins)	No of itera-tions	Con-vergence (mins)	No of itera-tions	Con-vergence (mins)	No of itera-tions
Minimum	0.60	12	0.33	13	1.47	12
Arbitrary	0.62	12	0.31	12	1.60	12
Maximum	0.62	12	0.35	13	1.85	12

Table (28) Convergence of NODEFLAN, MESHFLAN and LATTNODEFLAN for the three cases of overlap for the network due to Dolan

Choice of Overlap	Program					
	NODEFLAN		MESHFLAN		LATTNODEFLAN	
	Con-vergence (mins)	No of itera-tions	Con-vergence (mins)	No of itera-tions	Con-vergence (mins)	No of itera-tions
Minimum	0.89	12	0.39	13	2.54	12
Arbitrary	0.88	12	0.59	12	2.74	12
Maximum	0.88	12	0.5	13	3.20	12

Table (29) Showing the effect of initial estimate of branch flows for the maximum overlap case of TEST1 using NODEFLAN

Initial flow cu ft/min	No of iterations required for solution
0.001	13
0.01	12
1.0	11
100	12
10,000	13
100,000	13

Table (30) The number of iterations required for the maximum overlay case of TEST1 to converge to a variety of criterion for the programs NODEFLAN, MESHFLAN and HCMESHOUT

Convergence Percent of calculated flow	No of iterations for solution		
	NODEFLAN	MESHFLAN	HCMESHOUT
1	9	10	49
0.1	11	14	73
0.01	14	16	97
0.001	17	18	122
0.0001	20	21	164

Table (31) The effect on stability of NODEFLAN of a range of node to datum flows. Normal flows given in Table (2)

Node to Datum Flows	No of Iterations
0.01 of normal	15
0.1 of normal	10
normal	11
100 x normal	12

Table (32) The node to datum pressures obtained by NODEFLAN for the analysis of the minimum overlap case of TEST1

Node No	Pressure lb/ft^2
1	-5.41
2	-154.06
3	-208.77
4	-339.45
5	-185.71
6	-50.24
7	-793.24
8	79.33
9	-16.84
10	-39.07
11	-100.90
12	-100.76
13	-455.09
14	-491.41
15	-483.21
16	-491.26
17	-303.68
18	-340.49
19	-386.73
20	-523.82
21	-544.95

Table (33) The individual branch flows for NPSMESHFLAN with node 8 at a specified pressure and for the fully flow specified case of NODEFLAN

Branch No	NPSMESHFLAN Flow cu ft/min	NODEFLAN Flow cu ft/min
1	-48.52	-48.52
2	-38.69	-38.68
3	-30.84	-30.84
4	-69.52	-69.51
5	-54.38	-54.37
6	-48.74	-48.74
7	-46.92	-46.91
8	-46.34	-46.34
9	40.83	40.82
10	-72.80	-72.79
11	17.48	17.49
12	-38.14	-38.13
13	-50.34	-50.34
14	-117.31	-117.27
15	-40.52	-40.53
16	-84.01	-83.99
17	-45.55	-45.55
18	-37.55	-37.54
19	-31.48	-31.47
20	-90.08	-90.06
21	-1.81	-1.81
22	13.97	13.97
23	-88.42	-88.40
24	-50.87	-50.87
25	-82.35	-82.34

Table (33) (Continued)

Branch No	HPSMESHFLAN Flow cu ft/min	NODEFLAN Flow cu ft/min
26	-5.51	-5.51
27	-64.10	-64.09
28	-84.55	-84.54
29	-122.69	-122.66
30	-17.67	-17.67
31	35.15	35.15
32	62.51	62.49
33	-54.45	-54.44
34	-77.98	-77.97
35	-66.95	-66.95
36	-104.93	-104.91
37	-1.74	-1.74
38	-142.63	-142.60

Table (34) The node to datum pressures obtained by NODEFLAN in the analysis of the maximum overlap case of the program due to Knights and Allen

Node No	Pressure lb/ft^2
1	-59,656.6
2	-55,759.2
3	-11,125.8
4	-14,530.3
5	-40,721.8
6	-45,610.7
7	-38,200.6
8	64,765.6
9	65,308.1
10	53,409.7
11	16,906.4
12	-3,957.1
13	-17,374.6
14	-14,872.5
15	-52,357.9
16	26,836.9
17	-51,985.8
18	3,622.6
19	-101,281.7

Table (35) The analysis of the maximum overlap case of the Knights and Allen network for two different pressure specifications at the nodes compared to the flow fully specified case of NODEFLAN

Branch Number	NPSMES1FLAN		NODEFLAN
	Node 9 specified Flow cu ft/min	Nodes 9 and 11 specified Flow cu ft/min	Flow cu ft/min
1	-383.83	-383.82	-383.54
2	-255.57	-255.57	-255.46
3	-151.70	-151.70	-151.60
4	15.01	15.02	15.02
5	265.07	265.07	264.92
6	386.89	386.90	386.68
7	-109.16	-109.16	-109.16
8	314.36	314.36	314.16
9	-856.20	-856.19	-855.47
10	-112.70	-112.69	-112.70
11	220.68	220.67	220.47
12	-55.28	-55.23	-55.04
13	119.02	119.05	118.91
14	231.00	231.10	230.87
15	1080.72	1080.38	1080.38
16	-502.82	-502.81	-502.49
17	42.53	42.53	42.50
18	-92.21	-92.21	-92.20
19	-37.12	-37.12	-37.10
20	103.88	103.88	103.88
21	358.99	358.99	358.81

Table (35) (Continued)

Branch Number	NPSMESHFLAN		NODEFLAN
	Node 9 specified Flow cu ft/min	Nodes 9 and 11 specified Flow cu ft/min	Flow cu ft/min
22	278.20	278.20	278.07
23	134.72	134.72	134.68
24	669.45	669.46	669.32
25	349.66	349.97	349.97
26	-1446.97	-1446.94	-1446.30
27	-45.27	-45.27	-45.25
28	-111.99	-112.04	-111.98
29	997.10	997.06	996.32
30	-495.91	-495.91	-495.72
31	121.79	121.79	121.81
32	-76.58	-76.58	-76.56

Table (36) The connection list for the subdivided network TEST1

Branch No	Node No	Node No
	Cutsection 1	
1	-1	2
2	-2	3
3	-3	4
4	-2	4
5	-4	5
6	-1	5
7	5	-6
8	-1	6
9	-5	7
10	-7	22
	Cutsection 2	
1	-1	2
2	-1	3
3	-2	3
4	-3	22
	Cutsection 3	
1	-1	4
2	1	-2
3	-2	4
4	-2	3
5	3	-4
6	-4	11
7	4	-5
8	-5	11
9	10	-11

Table (36) (Continued)

Branch No	Node No	Node No
	Cutsection 3	
10	-9	10
11	-5	9
12	-5	7
13	5	-6
14	6	-7
15	-6	8
16	-7	8
17	-8	9
18	-8	22
	Cutbranches	
1	5	-9
2	6	-8
3	7	-10
4	-4	11
5	-3	11
6	-7	13

Table (37) The node to datum flows for the subdivided network TEST

Node No	Flow cu ft/min
	Cutsection 1
1	120
2	0
3	0
4	0
5	0
6	0
7	0
	Cutsection 2
1	210
2	0
3	0
	Cutsection 3
1	-120
2	0
3	0
4	0
5	0
6	0
7	-60
8	0
9	0
10	-240
11	0

Table (38) The results of the subdivided network TEST1 using NODEFLAN for comparison with the results of Middleton (6)

Branch Number	NODEFLAN	CASE 1 Middleton
	Cutsection 1 Flow cu ft/min	
1	88.40	88.34
2	50.87	50.88
3	31.47	31.74
4	37.54	37.56
5	83.99	83.93
6	45.55	45.56
7	48.52	48.53
8	13.97	14.98
9	48.74	48.76
10	1.74	1.74
	Cutsection 2	
1	69.51	69.48
2	77.97	77.96
3	30.84	30.86
4	54.37	54.38
	Cutsection 3	
1	46.91	46.91
2	5.51	5.53
3	46.34	46.33
4	40.83	40.83
5	64.09	64.09
6.	84.54	84.50
7	72.79	72.72
8	38.13	38.16
9	122.66	122.50

Table (38) (Continued)

Branch Number	NODEFLAN	CASE 1 Middleton
	Cutsection 3 Flow cu ft/min	
10	117.28	117.20
11	50.34	50.32
12	1.81	1.79
13	17.48	17.51
14	17.67	17.69
15	35.15	35.17
16	40.53	40.56
17	66.95	66.96
18	142.60	142.40
	Cutbranches	
1	38.68	38.70
2	62.48	62.46
3	54.43	54.45
4	90.04	89.98
5	82.31	82.25
6	104.87	104.80

Table (39) The node to datum pressures calculated by NODEDIAK for the network TEST1 shown in figure (5.4.1) and the equivalent results obtained by Middleton (6)

Node Number	Pressure lb/ft^2	
	NODEDIAK	Middleton
Cutsection 1		
1	95.36	95.32
2	-53.29	-53.04
3	-108.01	-107.72
4	-84.95	-84.68
5	50.52	50.52
6	100.76	100.73
7	-0.14	-0.14
Cutsection 2		
1	180.09	179.96
2	83.92	83.92
3	61.69	61.67
Cutsection 3		
1	-238.69	-238.14
2	-239.73	-239.19
3	-202.92	-202.40
4	-285.97	-285.38
5	-390.50	-389.63
6	-382.45	-381.56
7	-390.65	-389.78
8	-354.33	-353.42
9	-444.19	-443.25
10	-692.48	-690.92
11	-423.06	-422.21

Table (40) The connection list for the subdivided network of [illegible] and Allen

Branch Number	Node	Node
	Cutsection 1	
1	-1	2
2	-2	3
3	1	-4
4	4	-5
5	-4	20
6	-1	20
	Cutsection 2	
1	-1	3
2	-2	3
3	-3	4
4	-4	5
5	-2	20
	Cutsection 3	
1	-1	2
2	-2	3
3	-3	4
4	-1	4
5	-1	20
	Cutsection 4	
1	-1	2
2	-2	3
3	-3	4
4	-1	4

Table (40) (Continued)

Branch Number	Node	Node
	Cutsection 4	
5	-4	5
6	-1	5
7	-2	20
	Cutbranches	
1	-5	15
2	-5	19
3	-3	6
4	-2	7
5	-9	11
6	-10	11
7	-10	12
8	-14	17
9	-14	18

Table (41) The node to datum flows for the subdivided network due to Knights and Allen

Node Number	Flow cu ft/min
Cutsection 1	
1	-500.00
2	0
3	-166.67
4	-333.33
5	0
Cutsection 2	
1	-250.00
2	-250.00
3	0
4	0
5	-500.00
Cutsection 3	
1	3333.3
2	-666.67
3	-333.33
4	-833.33
Cutsection 4	
1	3166.7
2	-500.00
3	0
4	-1666.67
5	-500.00

Table (42) A comparison between the results of the analysis of the gas network due to Knights and Allen by Middleton and NODEDIAK

Branch Number		Flow cu ft/min		
NODEDIAK	Middleton	NODEDIAK	Middleton	Knights and Allen
	Cutsection 1			
1	5.1	104.65	108.36	105.53
2	4.1	151.68	150.58	151.35
3	5c	35.01	35.14	37.58
4	9.1	90.52	87.88	93.72
5	8.1	277.82	279.00	278.77
6	7.1	360.34	357.75	357.40
	Cutsection 2			
1	1.1	265.00	266.10	267.30
2	2.1	120.10	121.54	121.23
3	3c	385.10	391.60	389.78
4	14.2	108.42	105.74	109.07
5	6.1	386.23	374.45	388.25
	Cutsection 3			
1	10.2	851.49	856.06	851.22
2	9.2	108.05	110.58	107.22
3	7.2	225.29	222.35	227.75
4	8.2	1001.52	996.80	998.70
5	4c	671.98	690.00	669.22
	Cutsection 4			
1	15.2	1079.74	1079.86	1094.16
2	1.2	227.31	224.92	237.57
3	3.2	118.93	122.49	122.57
4	4.2	1453.60	1455.00	1456.43

Table (42) (Continued)

Branch Number		Flow cu ft/min		
NODEDIAK	Middleton	NODEDIAK	Middleton	Knights and Aller
	Cutsection 4			
5	6.2	42.84	43.51	38.10
6	5.2	502.56	502.43	459.00
7	2c	352.42	352.86	355.98
	Cutbranches			
1	10.1	130.81	128.90	174.77
2	1c	40.29	41.40	79.90
3	7c	14.99	16.68	16.08
4	3.1	256.32	259.18	258.42
5	13.2	493.50	496.87	500.30
6	12.2	314.81	315.54	313.45
7	11.2	76.77	78.71	77.42
8	2.2	108.38	111.97	113.57
9	6c	51.29	47.61	50.50

Table (43) The connection list for the cutsections of the network due to Dolan as shown in figure (5.4.3)

Branch Number	Node	Node
	Cutsection 1	
1	1	-13
2	-1	2
3	-2	3
4	-3	4
5	-4	5
6	-5	6
7	-6	8
8	-6	7
9	-8	9
10	-9	10
11	11	-13
12	-7	10
	Cutsection 2	
1	1	-13
2	-1	2
3	-2	3
4	-3	4
5	-3	6
6	5	-6
7	-6	7
8	-7	8
9	-7	12
10	-7	9
11	11	-12

Table (43) (Continued)

Branch Number	Node	Node
	Cutsection 2	
12	10	-11
13	-9	10
14	-5	8
15	-4	5
	Cutsection 3	
1	1	-13
2	-1	2
3	2	-3
4	-2	4
5	-4	5
	Cutbranches	
1	-8	20
2	-11	26
3	-12	28

Table (44) The dimensions of the network due to Dolan (32) with the cutsections as indicated in figure (5.4.3)

Branch Number	Length ft	Diameter ft
	Cutsection 1	
1	1500.0	1.0
2	5000.0	1.0
3	4100.0	1.0
4	3000.0	1.667
5	1000.0	1.333
6	3300.0	1.167
7	9300.0	1.0
8	8800.0	1.0
9	3300.0	1.0
10	2100.0	1.0
11	1000.0	0.667
12	3900.0	1.0
	Cutsection 2	
1	3400.0	1.0
2	2000.0	1.0
3	2200.0	1.0
4	5300.0	0.833
5	8000.0	1.333
6	1200.0	0.667
7	14100.0	1.333
8	2600.0	1.333
9	1200.0	1.0
10	4500.0	1.0
11	2000.0	1.0

Table (44) (Continued)

Branch Number	Length ft	Diameter ft
	Cutsection 2	
12	2000.0	0.5
13	3000.0	1.0
14	2500.0	0.667
15	4500.0	0.667
	Cutsection 3	
1	1000.0	0.5
2	2000.0	0.5
3	2500.0	0.833
4	1600.0	0.667
5	1500.0	0.667
	Cutbranches	
1	4000.0	1.0
2	5000.0	0.833
3	2200.0	0.833

Table (45) The node to datum supplies for the network due to Dolan shown in figure (5.4.3)

Node Number	Flow cu ft/min
Cutsection 1	
1	-6.25
2	-8.33
3	206.25
4	-2.083
5	-2.083
6	-6.25
7	-2.083
8	0
9	-20.3
10	-93.75
11	0
Cutsection 2	
1	0
2	-2.621
3	-4.167
4	0
5	-12.5
6	0
7	-2.083
8	0
9	0
10	-14.58
11	0
12	-6.25

Table (45) (Continued)

Node Number	Flow cu ft/min
Cutsection 3	
1	-4.167
2	0
3	-12.5
4	-2.083
5	-4.167

Table (46) The results of the NODEFLAN analysis of the minimum overlap case of Dolan's network shown in figure (5.2.11) and the NODEDIAK analysis for the subdivided network shown in figure (5.4.3) and the analysis of Ingels and Powers

Branch Number	Flow cu ft/min			
	NODEDIAK	Middleton	NODEFLAN	I and P
	Cutsection 1			
1	60.22	59.81	59.99	60.23
2	66.47	66.01	66.24	66.51
3	74.80	74.28	74.57	74.83
4	131.40	130.92	131.65	131.40
5	129.32	128.78	129.57	129.33
6	127.24	127.29	127.49	127.24
7	59.89	59.86	60.05	59.91
8	61.11	60.85	61.20	61.06
9	55.04	54.86	54.93	55.06
10	34.71	34.68	34.63	34.78
11	15.87	13.62	13.44	13.22
12	59.03	58.77	59.12	58.97
	Cutsection 2			
1	38.71	38.00	38.16	40.73
2	37.31	37.03	37.07	37.34
3	34.69	34.42	34.45	35.24
4	5.81	5.79	5.77	3.84
5	24.72	24.52	24.52	27.24
6	8.92	8.76	8.87	9.09
7	15.81	15.79	15.65	17.33

Table (46) (Continued)

Branch Number	Flow cu ft/min			
	NODEDIAK	Middleton	NODEFLAN	I and P
	Cutsection 2			
8	2.22	2.03	2.13	1.25
9	8.91	8.88	8.88	8.17
10	7.03	6.88	6.82	8.33
11	2.66	2.63	2.63	1.92
12	2.66	2.63	2.63	1.92
13	11.92	11.96	11.94	13.22
14	2.22	2.03	2.13	1.25
15	5.81	5.77	5.76	3.84
	Cutsection 3			
1	7.66	8.39	8.38	7.26
2	3.49	4.22	4.21	3.01
3	3.37	1.13	0.94	0.24
4	6.85	5.35	5.6	2.85
5	4.77	3.26	3.07	0.77
	Cutbranches			
1	15.86	13.62	13.44	13.21
2	4.69	5.09	5.12	4.32
3	0.61	0.90	1.09	3.39

Table (47) The connection list for the analysis of TEST1 by MESHDIAK, figure (5.4.4)

Branch Number	Node	Node
	Cutsection 1	
1	1	-19
2	-1	6
3	5	-6
4	-2	5
5	-3	5
6	4	-5
7	-4	7
8	-7	8
9	-7	9
10	9	-10
11	17	-18
12	16	-17
13	15	-16
14	12	-16
15	8	-12
16	11	-12
17	-12	14
18	-11	13
19	14	-17
20	13	-14
21	8	-11
22	4	-8
23	8	-9
24	-3	4

Table (47) (Continued)

Branch Number	Node	Node
	Cutsection 1	
25	-2	3
26	-1	2
27	12	-15
28	15	-17
29	6	-19
30	-6	10
31	-10	18
	Cutsection 2	
1	-1	7
2	-1	3
3	-3	6
4	2	-3
5	-2	4
6	-2	5
7	-1	2
8	6	-7
9	4	-6
10	-4	5

Table (48) The node to datum flows for the analysis of TE3T1 by MESHDIAK

Node Number	Flow cu ft/min
Cutsection 1	
1	120
2	0
3	0
4	-120
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	-240
14	0
15	-60
16	0
17	0
18	0
Cutsection 2	
1	90
2	0
3	0
4	0
5	210

Table (48) (Continued)

Node Number	Flow cu ft/min
Cutsection 2	
6	0
7	0

Table (49) The results obtained in the analysis of TEST1 by MESHDIAK compared with the results obtained for the maximum overlap case of the same network by MESHFLAN

Branch Number	Flow cu ft/min	
	MESHDIAK	MESHFLAN
	Cutsection 1	
1	13.94	13.97
2	45.41	45.55
3	83.68	84.01
4	37.43	37.54
5	31.37	31.48
6	89.73	90.08
7	5.50	5.51
8	46.18	46.34
9	40.69	40.83
10	104.50	104.93
11	142.06	142.64
12	35.05	35.15
13	17.63	17.66
14	17.45	17.48
15	72.51	72.81
16	38.02	38.13
17	50.16	50.34
18	122.19	122.69
19	66.74	66.96
20	116.83	117.30
21	84.24	84.56
22	46.76	46.92
23	63.88	64.10

Table (49) (Continued)

Branch Number	Flow cu ft/min	
	MESHDIAK	MESHFLAN
	Cutsection 1	
24	82.03	82.36
25	50.71	50.87
26	88.08	88.42
27	1.79	1.80
28	40.42	40.53
	Cutsection 2	
1	62.27	62.49
2	69.26	69.52
3	38.56	38.68
4	30.75	30.84
5	54.27	54.44
6	54.20	54.37
7	77.71	77.98
	Cutbranches	
1	48.45	48.52
2	48.68	48.75
3	1.73	1.74

Table (50) The node to datum flows for the network TEST2 as shown in figure (5.4.5). For reasons of space only non zero flows are indicated

Node Number	Flow cu ft/min
1	10
2	-10,000
3	-12,000
4	-5
5	-6
6	-3,000
7	-40
8	5,939
9	-10
16	100
17	1,000
24	-3,000
25	-2,000
32	5,000
33	10
39	-50
40	-100
47	50
48	900
55	-500
56	60
57	700
58	3,000
59	1,000

Table (50) (Continued)

Node Number	Flow cu ft/min
60	11,442
61	300
62	1,000
63	300

Table (51) The results obtained in the analysis of network TEST2 by NODEDIAK and MESHDIAK. The branch numbers refer to the analysis by NODEDIAK

Branch Number	NODEDIAK	MESHDIAK
	Flow cu ft/min	Flow cu ft/min
	Cutsection 1	
1	4059.09	4057.59
2	82.32	82.33
3	6088.15	6085.59
4	4049.09	4047.60
5	433.31	433.29
6	1086.06	1085.65
7	2913.43	2912.38
8	1187.13	1186.74
9	220.91	220.81
10	685.01	684.87
11	4382.40	4380.86
12	4067.43	4066.60
13	2154.14	2153.43
14	1072.33	1071.96
15	375.64	375.49
16	555.50	555.29
17	713.89	713.66
18	836.06	835.81
19	1221.68	1221.37
20	2047.70	2047.10
21	249.21	249.14
22	289.56	289.46
23	362.12	362.00

Table (51) (Continued)

Branch Number	NODEDIAK	MESHDIAK
	Flow cu ft/min	Flow cu ft/min
	Cutsection 1	
24	782.02	782.82
25	659.70	659.54
26	57.85	57.85
27	2246.95	2246.21
28	1289.12	1288.72
29	989.12	988.83
30	1656.75	1656.23
31	3652.24	3650.95
32	5858.36	5856.19
33	5993.88	5991.67
34	2731.44	2731.06
35	3116.86	3116.40
36	4166.53	4165.07
37	4339.01	4337.48
38	3433.10	3431.96
39	3200.32	3199.25
40	2878.72	2878.31
41	2460.43	2460.09
42	2503.63	2502.83
43	2351.29	2350.54
44	2229.11	2228.42
45	2345.25	2344.52
46	2393.00	2392.68
47	2612.26	2611.92

Table (51) (Continued)

Branch Number	NODEDIAK	MESHDIAK
	Flow cu ft/min	Flow cu ft/min
	Cutsection 1	
48	2417.81	2417.04
49	2269.46	2268.73
50	1667.62	1667.11
51	2295.49	2294.77
52	4317.00	4316.36
	Cutsection 2	
1	1711.01	1710.48
2	2848.16	2847.25
3	3158.06	3157.01
4	2780.83	2779.89
5	1398.46	1398.041
6	2026.29	2025.62
7	1911.98	1911.34
8	1271.09	1270.67
9	1017.78	1017.43
10	265.69	265.67
11	1372.37	1371.97
12	1361.89	1361.46
13	896.23	895.97
14	246.78	246.69
15	77.79	77.81
16	463.11	462.99
17	428.72	428.62
18	343.84	343.77

Table (51) (Continued)

Branch Number	NODEDIAK	MESHDIAK
	Flow cu ft/min	Flow cu ft/min
	Cutsection 2	
19	2465.63	2464.81
20	2171.79	2171.08
21	484.75	484.64
22	706.88	706.70
23	706.72	706.53
24	1155.57	1155.23
25	1297.07	1296.68
26	937.56	937.29
27	1787.04	1786.47
28	1349.48	1349.08
29	1289.48	1289.10
30	2213.82	2213.10
31	1647.48	1646.96
32	3372.52	3371.34
33	1656.75	1656.23
34	1862.77	1862.17
35	269.89	269.83
36	357.94	357.84
37	1748.46	1747.89
38	2272.78	2272.04
39	2426.96	2426.16
40	2001.77	2001.12
41	1641.40	1640.87
42	2290.85	2290.09

Table (51) (Continued)

Branch Number	NODEDIAK	MESHDIAK
	Flow cu ft/min	Flow cu ft/min
	Cutsection 2	
43	2326.34	2325.58
44	2391.83	2391.06
45	2366.87	2366.11
46	2291.94	2291.20
47	2205.98	2205.26
48	1983.85	1983.22
49	2292.10	2291.37
50	2492.20	2491.41
51	1624.33	1623.84
52	2433.60	2432.81
53	2725.02	2724.131
	Cutbranches	
1	3361.66	3360.58
2	2527.95	2527.13
3	1425.23	1424.79
4	42.66	42.69
5	488.06	487.92
6	581.37	581.22
7	922.73	922.50

Table (52) The node to datum flows for the network TEST3 shown in figure (5.4.8)

Node Number	Flow cu ft/min	Node Number	Flow cu ft/min
1	1000	24	-250
2	0	25	-600
3	-20	26	-90
4	-30	27	-100
5	-50	28	-760
6	2000	29	500
7	-100	30	90
8	-500	31	800
9	-600	32	-600
10	10	33	-25
11	75	34	-300
12	-800	35	-500
13	300	36	-900
14	-400	37	-75
15	500	38	-125
16	600	39	-350
17	700	40	400
18	800	41	-200
19	5	42	-200
20	7	43	0
21	8	44	0
22	-40	45	-985
23	-50		

Table (53) The results of the analysis of TEST3 by NODEFLAN with the node to datum flows as specified in table (52)

Branch Number	Flow cu ft/min	Branch Number	Flow cu ft/min
1	999.69	26	264.28
2	754.41	27	272.75
3	734.41	28	279.71
4	704.43	29	456.28
5	1843.48	30	167.52
6	622.35	31	1474.41
7	23.02	32	401.14
8	673.47	33	1195.84
9	126.33	34	748.49
10	828.94	35	146.64
11	1244.82	36	211.20
12	1755.17	37	211.20
13	585.84	38	215.00
14	245.33	39	184.94
15	195.34	40	199.97
16	922.58	41	839.82
17	1010.27	42	914.79
18	1809.91	43	1134.35
19	471.39	44	1389.59
20	387.24	45	967.25
21	726.89	46	1272.09
22	420.19	47	822.62
23	244.62	48	113.99
24	5.34	49	569.68
25	113.52	50	621.50

Table (53) (Continued)

Branch Number	Flow cu ft/min
51	655.36
52	1673.80
53	391.39
54	474.24
55	244.48
56	1107.02
57	287.80
58	706.76
59	611.07
60	378.75
61	1048.37
62	102.03
63	98.23
64	636.96
65	567.85

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
$\underline{0}$	Null matrix
$\underline{A}'$	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
$\underline{Jc}$, $\underline{Vc}$	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 cited in Appendices

For reasons of simplicity the references cited in the appendices are assigned the same reference numbers as those in the main text.

6 MIDDLETON P, 1968, Ph D Thesis, The University of Aston in Birmingham

13 ROTH J P, 1959, Quart App Math 17 No 1, April, p 1

19 BRAMIN F H, 1962, IBM Tech Rep TR 00 85 Data Systems Div Poughkeepsie NY., March

21 BRAMIN F H, 1967, Electronics 40 January, p 88

40 ABBOTT J M, 1971, Computer Users' Handbook, The University of Aston in Birmingham, November