STEADY STATE ANALYSIS OF PIPING NETWORKS

by

GEORGE C. GOODMAN

Submitted in partial fulfillment
of the requirements for the
degree of

BACHELOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 12, 1977

Signature of Author..
Department of Mechanical Engineering, May 12, 1977

Certified by..
Thesis Supervisor

Accepted by...
Chairman, Departmental Committee on Thesis

STEADY STATE ANALYSIS OF PIPING NETWORKS

by

GEORGE C. GOODMAN

Submitted to the Department of Mechanical Engineering on May 18, 1977 in partial fulfillment of the requirements for the degree of Bachelor of Science.

ABSTRACT

The author has developed a computer program to solve for the steady state response of arbitrary piping networks consisting of pipes, control valves and ideal pressure and flow sources. Recommendations are given for extension of the program to accomodate a greater variety of network elements. It is believed this program can easily be incorporated into a model of the energy interactions in the MIT Chilled Water Distribution System.

Thesis Supervisor: Richard S. Sidell
Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I would like to acknowledge the invaluable advice of Professor Richard Sidell who kept this effort going in a sensible direction and Gary Isaacs, whose energy and enthusiasm enabled me to bring this paper to a timely conclusion.

TABLE OF CONTENTS

I. INTRODUCTION

A project is under way at the Massachusetts Institute of Technology to implement centralized computer control of the environmental parameters in the academic and administrative buildings. Set points on thermostats, air vent openings, etc., will be modified according to the demands of building usage. The purpose of such control is, of course, to attempt to save energy. Achievement of this goal is threatened, however, by the following consideration.

The air conditioning systems in several buildings require chilled water as coolant. Water is chilled at a central plant where it is pumped through a three loop primary system. Each building has its own circulating pump to tap off the primary and is thus a secondary circulating system. The central plant is designed to operate at specified inlet and outlet temperatures and, naturally, runs at reduced efficiency when these temperatures vary from the ideal.

The proposed computer control system will not directly control the primary chilled water system. It will, however, indirectly control flow in the building secondary systems and thus tend to greatly vary the plant inlet temperature. This implies the plant will be running at lower efficiency and will, therefore, be wasting energy.

The next logical step is, therefore, to implement some form of control to maintain the desired plant inlet and outlet temperatures. The design of an effective control scheme requires a mathematical model of the energy interactions within the distribution system. As a first step in the development of such a model, this paper presents a FORTRAN computer program which allows the user to solve for the steady state pressures and flows in arbitrary piping networks. Section II explains the theory required to understand the program implementation described in Section III. At present, the only permissible network elements are pipes, control valves, and ideal constant pressure and flow sources. However, extension of the program to handle a larger variety of elements is straightforward and is discussed in Section IV. The FORTRAN source code and a sample network analysis are included in the Appendix.

II. THEORETICAL BACKGROUND

2.1 Pipes and Valves.

Mathematical approximations to the pressure-flow characteristics of pipes and control valves are well known. Assuming turbulent flow, the following equation has been used for pipes:

$$P = a_t \, Q|Q|^{3/4} \qquad (1)$$

where P is the pressure drop across the pipe, Q is the volume flow rate and a_t is a constant "whose value dependens on the flow rate at which transition to turbulent flow occurs, the dimensions of the pipe (diameter and length), the properties of the fluid (density and viscosity) and the roughness of the inner walls of the pipe."[1]

Since a control valve is nothing more than a variable orifice, the following equation for the pressure drop across an orifice has been used:

$$P = a_o \, Q|Q| \qquad (2)$$

where a_o is a function of the fluid density and the valve geometry.[2]

At present, the program requires the user to input both a_o and a_t explicitly. A simple modification could cause these constants to be calculated from parameters which might be more readily available.

2.2 Linear Graph Theory

The theory which follows attempts to define a procedure for deriving the network governing equations.

Piping networks lend themselves quite readily to representation as linear graphs. The following definitions and relationships apply to linear graphs in general.

Definition 1: Graph Edges and Vertices

"A linear graph is a set of line segments referred to as edges. End-point intersections of these edges are called vertices. All line segments are interconnected in such a way that the edges are only incident with the vertices."[3] The terms edge, branch, and element will be used interchangeably as will the terms vertex and node.

Definition 2: Circuit Loop

"If one begins at an initial vertex point of a graph and traverses through edges and vertices... until reaching the initial vertex without crossing any vertex more than once, the closed path traversed is called a circuit loop."[4] The terms circuit loop, circuit and loop will be used interchangeably.

Definition 3: Tree and Branch

"A tree T of some graph G consists of a subpart of the original graph. It contains all vertices of G but no circuits. The edges forming this subpart...are referred to as tree branches."[5] A given graph can have many possible trees.

Definition 4: Link

For a particular choice of tree, a link is an edge which is not one of the tree branches.

Definition 5: Fundamental Circuit

A fundamental circuit is defined with respect to a particular tree of a given graph. It is a circuit which contains one and only one link. For a given link there is a unique fundamental circuit.

It is easily shown that the number of links, L, and therefore the number of fundamental circuits, is given by

$$L = E - V + 1 \qquad (3)$$

where E is the number of edges in the graph and V is the number of vertices.

It is well known that an analogy may be drawn between piping networks and electrical networks where pressure corresponds to voltage and volume flow rate corresponds to current. Pursuing this analogy, Kirchhoff's Voltage

Law (KVL) and Kirchhoff's Current Law (KCL) may be applied.

If one were to write the KVL equations for all possible loops in a given network, it would soon be apparent that some of the equations are not independent. In fact, there are exactly L independent KVL equations. One set of such equations may be obtained by choosing any tree of the network and writing KVL for each of the resulting fundamental circuits. Obviously, only L of the variables in these equations are independent. Since the link pressures each appear only once in the set and since they all appear in different equations, they must be the L independent variables.

For non-linear networks, such as piping networks, it is not convenient to express the remaining tree pressures directly in terms of the link pressures. Rather, it is easier to calculate the tree pressures from the tree flows (by means of the elements' constitutive relations, assumed known), and to express the tree flows as linear combinations of the flows through the links. The link flows are therefore the independent variables in this set of L equations. These equations may be written in matrix form as follows:

$$Q = Kx \tag{4}$$

$$P = G(Q) \tag{5}$$

$$CP = 0 \tag{6}$$

where Q is an E x 1 vector containing the flows in all the elements, x is the L x 1 vector of link flows, and K is an E x L matrix of constants. Equation (4) expresses the tree flows as linear combinations of the link flows. In Equation (5), P is an E x 1 vector containing the pressure drops across all the elements and G is a vector function containing the constitutive relations for all elements. Matrix C is an L x E matrix of constants. Equation (6) is the explicit statement of KVL.

It should be fairly obvious that the only constants C can contain must be 1 , 0 or -1. Similarly, K also contains only those constants. In fact, it can be shown that K is the transpose of C. The matrix K therefore contains all the topological information necessary to derive the network governing equations. The problem of deriving the network equations has been solved except for one detail. The K matrix representation of network topology is not a representation most people would find convenient to use. Another more intuitive representation is therefore proposed.

Let us construct a V x E matrix D according to the following procedure. Number the elements and the nodes of the matrix under consideration. Now, assign arbitrary directions to the flows through all the elements. This defines a starting and ending node for each branch.

Label the columns of D with the edge numbers and the rows with the vertex numbers. Now, considering each column and its associated element in turn, place a 1 in the row corresponding to the starting node of the element and a -1 in the ending node row.

This matrix is called the incidence matrix for the network and contains the same topological information as the K matrix. It follows then, that K may be derived from D. That derivation will now be described.

First, since every column of D has exactly one 1 and one -1 in it, one row of the matrix may be removed without losing any topological information. The resulting (V - 1) x E matrix is called the incidence sub-matrix, A.

Identify a set of tree branches for the network. Any set of (V - 1) linearly independent columns of D is one such set.[6] Now define

$$Q_f = A_t^{-1} A \qquad (7)$$

where A_t is a (V - 1) x (V - 1) matrix containing the above mentioned tree columns. Rearrange the columns of Q_f so that the resulting matrix may be partitioned as follows:

$$Q_f = [Q_{f_{11}} | U_g] \quad (8)$$

where U_g is a (V - 1) x (V - 1) identity matrix. Note, the resulting column order is different from the column order of the A matrix.

Now define:

$$B_{f_{12}} = -Q^T_{f_{11}} \quad (9)$$

Construct matrix B_f,

$$B_f = [U_b | B_{f_{12}}] \quad (10)$$

where U_b is an L x L identity matrix. Matrix B_f is called the fundamental circuit matrix and is the desired K matrix with the elements renumbered in the same order as Q_f[6].

2.3 Non-linear Equations.

This section presents an algorithm for the solution of a set of simultaneous non-linear equations.

Consider a vector of functions F whose argument is the vector x. We wish to find the value of x which satisfies the equation

$$F(x) = 0 \quad (11)$$

Expanding (11) in a Taylor series about the desired solution vector x_o and neglecting higher order terms yields

$$F(x_o) + H(x_o)\ (x - x_o) = 0 \qquad (12)$$

where H is a square matrix such that

$$H_{ij} = \frac{\partial F_i}{\partial x_j} \qquad (13)$$

If we assume that x_n is the nth iteration of an iterative algorithm, the following recursion relation may be deduced

$$x_{n+1} = -(H(x_n))^{-1}F(x_n) + x_n \qquad (14)$$

The derivative H_{ij} may be approximated by perturbing the jth variable an amount P to give the vector x^*_j and setting

$$H_{ij} = \frac{F_i\ (x^*_j) - F_i(x)}{P} \qquad (15)$$

Iteration stops when $F(x_n)$ is less than a prescribed tolerance.

III. IMPLEMENTATION

This section will explain the actual program structure and operation.

3.1 Structure.

The main program is responsible for the iterative solution of the L network governing equations. It uses the algorithm presented in Section 2.3 with x equal to the vector of link flows and F the function such that F_i is the sum of the pressure drops around the ith fundamental circuit. A subroutine named F is called each time evaluation of the function is required.

Before iteration of the solution algorithm can begin, the fundamental circuit matrix, here called B, must be constructed. The subroutine INIT is called which accepts user input and formulates the B matrix according to the procedure of Section 2.2.

When the solution has been found the subroutine OUTPUT is called which prints the results.

3.2 Operation.

A data card for the program looks like this:

cc 1-10 - User pipe number (integer)
array = USERP

cc 11-20 - User starting node number (integer)
array = USER

cc 21-30 - User ending node number (integer)
array = USER

cc 31-40 - Element type number (integer)
array = NTYPE

1 = pipe
2 = pressure source
3 = flow source
4 = control valve

cc 41-50 - Appropriate datum (real)
array = PIPE

cc 51-60 - Parameter (real)
array = PRMTR

not currently used

cc 61-70 - X - coordinate (real)
array = COORDS (I,1)

not currently used

cc 71-80 - Y - coordinate (real)
array = COORDS (I,2)

not currently used

A card with a zero in cc 1 indicates the end of the data.

Subroutine INIT reads the data and stores it in appropriate arrays. Each time it reads a card it adds a column to the incidence matrix A. If the user node numbers are new numbers, it adds appropriate rows to the matrix and makes the correct entries. The element types are then examined for flow sources. The columns corresponding to flow sources are shifted to the rightmost columns in the matrix. The other stored data is also sorted the same way to maintain internal pipe numbering consistency. The incidence matrix is then transposed and passed to the subroutine REDUCE.

The function of REDUCE is to identify the subscripts of (V-1) linearly independent columns of A. These columns are used to fill the matrix A_t . The product $A_t^{-1} A$ is computed. The columns representing an identity sub-matrix are identified and rearranged to give $[Q_{f11} | U_g]$. The stored element data is sorted again according to this rearrangement.

The B_f matrix is constructed. The array of element types is searched again for flow sources. The last (V-1) elements of the rows corresponding to fundamental circuits containing flow sources are shifted to the bottom of B. The first L elements of stored edge data are sorted according to this last shifting.

The MAIN program now treats the problem as if the number of independent variables is L minus the number of flow sources. Subroutine F calculates pressure drops across flow sources in such a way as to guarantee satisfaction of KVL.

IV. RECOMMENDATIONS

Several dummy subroutines are called which, when replaced with actual subroutines, allow the extension of this program to accept a larger variety of network elements.

The function ASSIGN can be used to retrieve the number of a data set in which tabulated pressure-flow data may be stored. Subroutine F can then call PRESS to evaluate the necessary constitutive relations.
The COMPUTED GO TO statement in subroutine F can be altered to accomodate elements having a known closed form constitutive relation.

V. CONCLUSIONS

This program makes the calculation of steady state, DC response of piping networks an easy matter. It should be a straightforward task to incorporate this program in the proposed energy interaction model. Minor additions allow the extension of the program to a greater variety of network elements. The accuracy of the assumed constitutive relations is, of course, still a question to be experimentally verified.

REFERENCES

1. Shearer, Murphy, Richardson, Introduction to System Dynamics, Addison-Wesley, 1967.

2. Davis, Palmer, Computer Aided Analysis of Electrical Networks, Charles E. Merrill, 1973.

FOOTNOTES

1. Shearer, Murphy, Richardson, Introduction to System Dynamics, p. 67.

2. Ibid., pp. 67-68.

3. Davis, Palmer, Computer Aided Analysis of Electrical Networks, p. 124.

4. Ibid., pp. 124-125.

5. Ibid., p. 126.

6. Ibid., pp. 285-286.

7. Ibid., pp. 123-172.

APPENDIX I. FORTRAN CODE & SAMPLE OUTPUT

```
C   PROGRAM:       MAIN
C
C   ARGUMENTS:     NONE
C
C   CALLS:         INIT, OUTPUT, INVERS, EXIT
C
C   DESCRIPTION:   THIS IS A PROGRAM TO PERFORM
C                  STEADY STATE ANALYSIS OF PIPING
C                  NETWORKS.  IT CALLS INIT TO INITIAL-
C                  IZE THE NECESSARY MATRICES, THEN
C                  USES AN ITERATIVE TECHNIQUE TO
C                  SOLVE THE RESULTING NON-LINEAR
C                  EQUATIONS.  THE SOLUTION IS CON-
C                  SIDERED COMPLETE WHEN THE ERROR
C                  IS LESS THAN THE VALUE OF THE
C                  TOLERANCE, TOL.
C
C
C
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 E,NTYPE(30)
      REAL P(30),Q(30),B(30,30),PIPE(30)
      INTEGER*2 USERP(30)
      REAL FO(30),DELTA(30),FXP(30),DFDX(30,30),L1(30),M1(30),DX(30)
      REAL X(30)
      REAL TEMP(900)
      COMMON E,L,P,Q,B,NTYPE,PIPE,NSRCE
      COMMON/BLK1/USERP
      COMMON/BLK2/X
C                 READ THE DATA AND INITIALIZE THE
C                 REQUIRED MATRICES
      CALL INIT
```

```
      LL=L-NSRCE
      PERT=.1
      TOL=.001
      IDIM=30
50    CONTINUE
      NUM=0
C                FO=F(X)
      CALL F(FO,X)
C                COUNT THE DELTAS LESS THAN TOL
      DO 100 I=1,LL
      DELTA(I)=-FO(I)
      IF(ABS(DELTA(I)).LE.TOL)NUM=NUM+1
100   CONTINUE
C                ARE WE FINISHED?
      IF(NUM.EQ.LL)CALL OUTPUT
C                CALCULATE DFDX, THE MATRIX OF
C                PARTIAL DERIVATIVES
      DO 200 J=1,LL
      X(J)=X(J)+PERT
      CALL F(FXP,X)
      DO 150 I=1,LL
      DFDX(I,J)=(FXP(I)-FO(I))/PERT
150   CONTINUE
      X(J)=X(J)-PERT
200   CONTINUE
C                DFDX=INVERSE OF DFDX
      CALL INVERS(DFDX,LL,IDIM,TEMP,L1,M1)
C                INCREMENT THE LINK FLOW VECTOR BY
C                DX WHICH IS THE PRODUCT OF THE
C                MATRIX DFDX AND THE VECTOR DELTA.
      DO 300 I=1,LL
      DX(I)=0.
      DO 250 J=1,LL
```

```
250     DX(I)=DX(I)+DFDX(I,J)*DELTA(J)
        X(I)=X(I)+DX(I)
300     CONTINUE
C                    GO BACK FOR ANOTHER ITERATION
        GO TO 50
        CALL EXIT
        END
PROGRAM   *MAIN* HAS     NO ERRORS
```

USER=GOODMAN 274 84255   JOINT COMPUTER FACILITY, MIT

```
C   PROGRAM:        INIT
C
C   ARGUMENTS:      NONE
C
C   CALLS:          ICSORT, ISORT, SORT, REDUCE, ASSIGN
C                   INVERS, ERROR1,ERROR3
C
C   DESCRIPTION: THIS PROGRAM READS THE INPUT DATA
C                AND FROM THAT DATA CONSTRUCTS THE
C                FUNDAMENTAL CIRCUIT MATRIX FOR THE
C                NETWORK UNDER ANALYSIS.
C
C
C
      SUBROUTINE INIT
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 E,NTYPE(30)
      REAL P(30),Q(30),B(30,30),PIPE(30)
      INTEGER*2 USERP(30)
      REAL X(30)
      INTEGER*2 V,A(30,30),SUB1(30),USER(30)
      REAL M1(30),M2(30)
      INTEGER*2 V1
      INTEGER*2 QF(30,30)
      REAL TEMP(900)
      REAL PRMTR(30)
      REAL COORDS(30,2)
      INTEGER*2 SUB(30)
      COMMON E,L,P,Q,B,NTYPE,PIPE,NSRCE
      COMMON/BLK1/USERP
      COMMON/BLK2/X
C
      IDIM=30
```

```
      E=0
      V=0
C                INITIALIZE THE SUBSCRIPT VECTORS
C                SUB AND SUB1.
C                INITIALIZE B AND A TO ZERO.
      DO 7 I=1,IDIM
      SUB(I)=I
      SUB1(I)=I
      DO 7 J=1,IDIM
      B(I,J)=0.0
7     A(I,J)=0
5     CONTINUE
C                READ THE DATA
      READ(8,1001)IPIPE,NSTART,NEND,ITYPE,DATUM,PARAM,XCOORD,YC
1001  FORMAT(4I10,4F10.4)
C                IS THIS THE LAST DATA CARD?
      IF(IPIPE.EQ.0)GO TO 50
      E=E+1
C                STORE THE ELEMENT DATA
      PIPE(E)=DATUM
C                ELEMENT TYPES GREATER THAN 4 ARE
C                SPECIAL CASES AND MAY REQUIRE DIF-
C                FERENT VALUES TO BE STORED IN PIPE.
      IF(ITYPE.GT.4)PIPE(E)=ASSIGN(ITYPE)
      NTYPE(E)=ITYPE
      USERP(E)=IPIPE
      COORDS(E,1)=XCOORD
      COORDS(E,2)=YCOORD
      PRMTR(E)=PARAM
C
C                CONSTRUCT THE INCIDENCE MATRIX
      DO 10 I=1,V
      IF(USER(I).NE.NSTART)GO TO 10
```

```
      A(I,E)=1
      GO TO 20
10    CONTINUE
C
      V=V+1
      USER(V)=NSTART
      A(V,E)=1
20    CONTINUE
C
      DO 30 I=1,V
      IF(USER(I).NE.NEND)GO TO 30
      A(I,E)=-1
      GO TO 5
30    CONTINUE
C
      V=V+1
      USER(V)=NEND
      A(V,E)=-1
      GO TO 5
50    CONTINUE

      IF(E.EQ.0)CALL ERROR1
      V1=V-1
      NSRCE=0
      NON=0
C
C                REARRANGE THE SUBSCRIPT VECTOR SO
C                THAT SOURCES APPEAR AS THE RIGHTMOST
C                ELEMENTS
      DO 60 I=1,E
      IF(NTYPE(I).EQ.3)GO TO 53
      NON=NON+1
      SUB1(NON)=I
```

```
      GO TO 60
53    CONTINUE
      ISUB=E-NSRCE
      SUB1(ISUB)=I
      NSRCE=NSRCE+1
60    CONTINUE
C
C                 SORT THE COLUMNS OF THE A MATRIX
C                 AND ALL THE DATA VECTORS ACCORDING
C                 TO THE ORDER OF THE SUBSCRIPTS IN
C                 SUB1.
      CALL ICSORT(A,V,E,SUB1)
      CALL SORT(PIPE,E,1,SUB1)
      CALL ISORT(NTYPE,E,1,SUB1)
      CALL ISORT(USERP,E,1,SUB1)
      CALL SORT(COORDS,E,2,SUB1)
      CALL SORT(PRMTR,E,1,SUB1)
C                 SET B=TRANSPOSE OF A
      DO 70 I=1,E
      DO 70 J=1,V
70    B(I,J)=A(J,I)
C
C                 ROW REDUCE B TO FIND THE INDEPEN-
C                 DENT COLUMNS OF A.
      CALL REDUCE(B,E,V,SUB1)
C
C                 FILL B WITH THE INDEPENDENT COLUMNS
C                 OF A.  THESE REPRESENT THE TREE
C                 BRANCHES FOR THE NETWORK.
      DO 80 J=1,V1
      ISUB=SUB1(J)
      DO 80 I=1,V1
80    B(I,J)=A(I,ISUB)
```

```
C
C                 B=INVERSE OF B
      CALL INVERS(B,V1,IDIM,TEMP,M1,M2)
C
C                 QF=PRODUCT OF MATRICES B AND A
      DO 90 I=1,V1
      DO 90 J=1,E
      QF(I,J)=0
      DO 90 K=1,V1
      QF(I,J)=QF(I,J)+IFIX(B(I,K))*A(K,J)
90    CONTINUE
C
C                 FIND THE IDENTITY SUBMATRIX IN QF.
      L=E-V1
      IV2=V-2
C
      DO 91 J=1,E
91    SUB1(J)=J
C
      DO 100 I=1,V1
C
      DO 95 JJ=1,E
      J=SUB1(JJ)
      IF(QF(I,J).NE.1)GO TO 95
      ICOUNT=0
C
      DO 93 II=1,V1
      IF(QF(II,J).EQ.0)ICOUNT=ICOUNT+1
93    CONTINUE
C
      IF(ICOUNT.NE.IV2)GO TO 95
      ISUB=L+I
      ISAVE=SUB1(ISUB)
```

```
      SUB1(ISUB)=J
      SUB1(JJ)=ISAVE
      GO TO 100
95    CONTINUE
C
      CALL ERROR3
100   CONTINUE
C
C                 CONSTRUCT THE FUNDAMENTAL CIRCUIT
C                 MATRIX, B.
      DO 110 I=1,L
      DO 110 J=1,L
      B(I,J)=0
      IF(I.EQ.J)B(I,J)=1
110   CONTINUE
C
      L1=L+1
C
      DO 120 I=1,L
      ISUB=SUB1(I)
      DO 120 J=L1,E
      KSUB=J-L
120   B(I,J)=-QF(KSUB,ISUB)
C
C                 SORT THE ELEMENT DATA, CONSISTENT
C                 WITH THE ORDERING OF THE COLUMNS
C                 OF QF.
      CALL SORT(PIPE,E,1,SUB1)
      CALL ISORT(NTYPE,E,1,SUB1)
      CALL ISORT(USERF,E,1,SUB1)
      CALL SORT(COORDS,E,2,SUB1)
      CALL SORT(PRMTR,E,1,SUB1)
C
```

```
C                 SEARCH THE ELEMENT TYPES FOR THE
C                 FLOW SOURCES
      NSRCE=0
      NON=0
      DO 130 I=1,L
      IF(NTYPE(I).EQ.3)GO TO 125
      NON=NON+1
      SUB1(NON)=I
      GO TO 130
125   CONTINUE
      ISUB=L-NSRCE
       SUB1(ISUB)=I
      NSRCE=NSRCE+1
130   CONTINUE

C                 RENUMBER THE LINK FLOWS SO THAT
C                 THE FLOW SOURCES HAVE THE HIGHEST
C                 SUBSCRIPTS.  INTERCHANGE THE APPRO-
C                 PRIATE ROWS OF B AND SORT THE ELEMENT
C                 DATA VECTORS.
      DO 135 I=1,L
      DO 135 J=L1,E
      K=J-L
135   QF(I,K)=B(I,J)
      CALL ISORT(QF,L,V1,SUB1)
      DO 137 I=1,L
      DO 137 J=L1,E
      K=J-L
137   B(I,J)=QF(I,K)

      CALL SORT(PIPE,L,1,SUB1)
      CALL ISORT(NTYPE,L,1,SUB1)
      CALL ISORT(USERP,L,1,SUB1)
```

```
      CALL SORT(COORDS,L,2,SUB1)
      CALL SORT(PRMTR,L,1,SUB1)
      DO 140 I=1,L
      X(I)=1.0
      IF(NTYPE(I).EQ.3)X(I)=PIPE(I)
140   CONTINUE

      RETURN
      END
PROGRAM  INIT  HAS   NO ERRORS
```

```
C   PROGRAM:      F
C
C   ARGUMENTS:    A - THE VECTOR EQUAL TO F(X)
C                 X - THE VECTOR OF LINK FLOWS
C   CALLS:        PRESS, ABS
C
C   DESCRIPTION: SUBROUTINE F EVALUATES THE VECTOR
C                 FUNCTION F(X) WHICH IS THE SUM OF
C                 THE PRESSURES AROUND THE FUNDAMENTAL
C                 CIRCUITS
C
      SUBROUTINE F(A,X)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 E,NTYPE(30)
      REAL P(30),Q(30),B(30,30),PIPE(30)
      REAL X(30),A(30)
      INTEGER*2 SUB(30)
      COMMON E,L,P,Q,B,NTYPE,PIPE,NSRCE

C                 Q=PRODUCT OF MATRICES B TRANSPOSE
C                 AND X.  Q IS THE VECTOR OF FLOWS
C                 THROUGH ALL THE ELEMENTS OF THE
C                 NETWORK.
      LL=L
      DO 10 I=1,E
      Q(I)=0.
      DO 10 K=1,LL
10    Q(I)=Q(I)+B(K,I)*X(K)

C                 CALCULATE THE PRESSURE DROP ACROSS
C                 ALL THE ELEMENTS FROM THE VECTOR
C                 OF FLOWS.  THE PROCEDURE IS DIFFER-
C                 ENT FOR EACH TYPE OF ELEMENT.
```

```
      PEXP=.75
      NSRCE=0
      DO 100 I=1,E
      QQ=Q(I)
      ITYPE=NTYPE(I)
      IF(ITYPE.GT.4)GO TO 30
      GO TO (40,50,60,70),ITYPE
30    CALL PRESS(I)
      GO TO 100
40    P(I)=PIPE(I)*QQ*ABS(QQ)**PEXP
      GO TO 100
50    P(I)=PIPE(I)
      GO TO 100
60    NSRCE=NSRCE+1
      SUB(NSRCE)=I
      P(I)=0.
      GO TO 100
70    P(I)=PIPE(I)*QQ*ABS(QQ)
100   CONTINUE

C                 CALCULATE PRESSURE DROP AROUND
C                 FUNDAMENTAL CIRCUIT LOOPS
      DO 105 I=1,LL
      A(I)=0.
      DO 105 K=1,E
105   A(I)=A(I)+B(I,K)*P(K)
      IF(NSRCE.EQ.0)RETURN

C                 SET THE PRESSURE DROP ACROSS ALL
C                 FLOW SOURCES SO THAT KVL IS SATIS-
C                 FIED.
      DO 110 I=1,NSRCE
      ISUB=SUB(I)
```

USER=GOODMAN 274 84255   JOINT COMPUTER FACILITY, MIT

```
110   P(ISUB)=-A(ISUB)
      RETURN
      END
PROGRAM  F      HAS    NO ERRORS
```

```
USER=GOODMAN 274 84255     JOINT COMPUTER FACILITY, MIT

C   PROGRAM:       PRESS
C
C   ARGUMENTS:     I - THE SUBSCRIPT IDENTIFYING AN
C                  ELEMENT WHOSE PRESSURE DROP IS TO
C                  BE CALCULATED.
C
C   CALLS:         NOTHING
C
C   DESCRIPTION: PRESS IS A DUMMY SUBROUTINE TO CAL-
C                  CULATE THE PRESSURE DROP ACROSS
C                  EXOTIC NETWORK ELEMENTS.
C
C
C
        SUBROUTINE PRESS(I)
        IMPLICIT INTEGER*2 (I-N)
        INTEGER*2 E,NTYPE(30)
        REAL P(30),Q(30),B(30,30),PIPE(30)
        COMMON E,L,P,Q,B,NTYPE,PIPE
        P(I)=20.
        RETURN
        END
PROGRAM  PRESS  HAS    NO ERRORS
```

```
USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

C   PROGRAM:       OUTPUT
C
C   ARGUMENTS:     NONE
C
C   CALLS:         EXIT
C
C   DESCRIPTION: OUTPUT PRINTS OUT THE DESIRED
C                RESULTS, THEN CALLS EXIT.
C
C
C
      SUBROUTINE OUTPUT
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 E,NTYPE(30)
      REAL P(30),Q(30),B(30,30),PIPE(30)
      INTEGER*2 USERP(30)
      COMMON E,L,P,Q,B,NTYPE,PIPE
      COMMON/BLK1/USERP
      WRITE(5,1000)
1000  FORMAT(//,6X,'PIPE NO.',15X,'PRESSURE',15X,'FLOW RATE',/)
      DO 100 I=1,E
100   WRITE(5,1010)USERP(I),P(I),Q(I)
1010  FORMAT(1X,I10,15X,F10.4,15X,F10.4)
      CALL EXIT
      END
PROGRAM  OUTPUT HAS    NO ERRORS
```

```
C   PROGRAM:       REDUCE
C
C   ARGUMENTS:    A - A MATRIX TO BE ROW REDUCED
C                  M - NUMBER OF ROWS IN A
C                  N - NUMBER OF COLUMNS IN A
C                  SUB - VECTOR OF SUBSCRIPTS INDICAT-
C                        ING THE REARRANGEMENT OF A
C                        DURING REDUCTION.
C
C   CALLS:         NOTHING
C
C   DESCRIPTION:  REDUCE PERFORMS ROW REDUCTION ON
C                  ARGUMENT MATRIX A.  THE VECTOR SUB
C                  CONTAINS THE ORIGINAL A MATRIX SUB-
C                  SCRIPTS IN THE FINAL ORDER.  REDUCE
C                  ONLY REDUCES THE MATRIX BELOW THE
C                  DIAGONAL.
C
C
C
      SUBROUTINE REDUCE(A,M,N,SUB)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SUB(M)
      REAL A(30,30)
      DO 10 I=1,M
10    SUB(I)=I
C
      N1=N-1
      DO 100 K=1,N1
C
      DO 20 I=K,M
      IF(A(I,K).EQ.0.0)GO TO 20
      PIVOT=A(I,K)
```

```
      ISUB=I
      GO TO 30
20    CONTINUE
C
      CALL ERROR4
30    CONTINUE
C
      DO 40 J=1,N
      HOLD=A(ISUB,J)
      A(ISUB,J)=A(K,J)
40    A(K,J)=HOLD
C
      ISAVE=SUB(ISUB)
      SUB(ISUB)=SUB(K)
      SUB(K)=ISAVE
      IF(K.EQ.N1)RETURN
      IF(PIVOT.EQ.1.0)GO TO 50
C
      DO 45 J=1,N
45    A(K,J)=A(K,J)/PIVOT
C
50    CONTINUE
      K1=K+1
C
      DO 60 I=K1,M
      UMULT=A(I,K)
      DO 60 J=K,N
60    A(I,J)=A(I,J)-UMULT*A(K,J)
C
100   CONTINUE
C
      END
PROGRAM  REDUCE HAS    NO ERRORS
```

```
C   PROGRAM:      INVERS
C
C   ARGUMENTS:    A - MATRIX TO BE INVERTED
C                 N - NUMBER OF ROWS USED IN A
C                 K - NUMBER OF ROWS AND COLUMNS TO
C                     WHICH A IS DIMENSIONED IN THE
C                     CALLING PROGRAM.
C                 B - A WORKING VECTOR OF DIMENSION
C                     K SQUARED OR LARGER.
C                 M1 - WORKING VECTOR OF LENGTH K
C                 M2 - WORKING VECTOR OF LENGTH K
C
C   CALLS:        MINV, ABS, ERROR2
C
C   DESCRIPTION:  INVERS FINDS THE INVERSE OF THE
C                 MATRIX A.  IT IS USEFUL WHEN THE
C                 MATRIX TO BE INVERTED ONLY CONTAINS
C                 USEFUL DATA IN THE UPPER N BY N
C                 SUB-MATRIX.  INVERS PACKS THE MATRIX
C                 THEN CALLS THE SSP ROUTINE MINV TO
C                 DO THE ACTUAL INVERSION.  THE RESULT
C                 IS STORED IN A, THEREFORE THE
C                 ORIGINAL IS DESTROYED.
C
C
C
      SUBROUTINE INVERS(A,N,K,B,M1,M2)
      IMPLICIT INTEGER*2 (I-N)
      REAL A(K,K),B(N,N)
      REAL M1(N),M2(N)
      DO 10 I=1,N
      DO 10 J=1,N
10    B(I,J)=A(I,J)
```

```
      CALL MINV(B,N,DET,M1,M2)
      IF(ABS(DET).LT..0001)CALL ERROR2
      DO 20 I=1,N
      DO 20 J=1,N
20    A(I,J)=B(I,J)
      RETURN
      END
PROGRAM  INVERS HAS    NO ERRORS
```

```
USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

C  PROGRAM:       ASSIGN
C
C  ARGUMENTS:     N - NUMBER OF ELEMENT FOR WHICH
C                 A VALUE OF PIPE MUST BE ASSIGNED.
C
C  CALLS:         NOTHING
C
C  DESCRIPTION:    ASSIGN IS A DUMMY FUNCTION.  A
C                 FUNCTIONAL VERSION WOULD RETURN A
C                 NUMBER, USUALLY A DATA SET NUMBER,
C                 TO BE STORED IN THE DATA VECTOR,
C                 PIPE.  ASSIGN WOULD BE RESPONSIBLE
C                 FOR INTERPRETING EXOTIC USER
C                 ELEMENT TYPES.
C
C
C
      FUNCTION ASSIGN(N)
      ASSIGN = 75
      RETURN
      END
PROGRAM  ASSIGN HAS    NO ERRORS
```

USER=GOODMAN 274 84255 JOINT COMPUTER FACILITY, MIT

```
      SUBROUTINE ERROR1
      WRITE(5,1000)
1000  FORMAT(//,' NO PIPES HAVE BEEN ENTERED',//)
      CALL EXIT
      END
```

PROGRAM ERROR1 HAS NO ERRORS

```
        USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

              SUBROUTINE ERROR2
              WRITE(5,1000)
        1000  FORMAT(//,' INCIDENCE MATRIX TREE COLUMNS WERE DEPENDENT'
              CALL EXIT
              END
PROGRAM  ERROR2 HAS    NO ERRORS
```

USER=GOODMAN 274 84255   JOINT COMPUTER FACILITY, MIT

```
      SUBROUTINE ERROR3
      WRITE(5,1000)
1000  FORMAT(//,' UNABLE TO IDENTIFY IDENTITY SUBMATRIX',//)
      CALL EXIT
      END
PROGRAM  ERROR3 HAS    NO ERRORS
```

```
USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

      SUBROUTINE ERROR4
      WRITE(5,1000)
1000  FORMAT(//,' NO NON-ZERO PIVOT FOUND',//)
      CALL EXIT
      END
PROGRAM  ERROR4 HAS    NO ERRORS
```

```
USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

C   PROGRAM:       ISORT
C
C   ARGUMENTS:     A - MATRIX TO BE SORTED
C                  M - NUMBER OF ROWS IN A
C                  N - NUMBER OF COLUMNS IN A
C                  SUB - VECTOR CONTAINING SUBSCRIPTS
C                        IN THE DESIRED ORDER FOR THE
C                        ROWS OF A.
C
C   CALLS:         NOTHING
C
C   DESCRIPTION:   ISORT (INTEGER SORT) IS A PROGRAM
C                  TO SORT THE ROWS OF AN INTEGER
C                  MATRIX, A, IN THE ORDER DICTATED
C                  BY THE VECTOR OF SUBSCRIPTS, SUB.
C
C
C
      SUBROUTINE ISORT(A,M,N,SUB)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 A(30,30),B(30,30)
      INTEGER*2 SUB(M)
      DO 10 I=1,M
      ISUB=SUB(I)
      DO 10 J=1,N
10    B(I,J)=A(ISUB,J)
      DO 20 I=1,M
      DO 20 J=1,N
20    A(I,J)=B(I,J)
      RETURN
      END
PROGRAM  ISORT  HAS    NO ERRORS
```

```
USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

C   PROGRAM:       SORT
C
C   ARGUMENTS:     A - MATRIX TO BE SORTED
C                  M - NUMBER OF ROWS IN A
C                  N - NUMBER OF COLUMNS IN A
C                  SUB - VECTOR OF SUBSCRIPTS
C
C   CALLS:         NOTHING
C
C   DESCRIPTION: SORT SORTS THE ROWS OF THE MATRIX
C                  A IN THE ORDER DICTATED BY THE
C                  VECTOR SUB.
C
C
C
      SUBROUTINE SORT(A,M,N,SUB)
      IMPLICIT INTEGER*2 (I-N)
      REAL A(30,30),B(30,30)
      INTEGER*2 SUB(M)
      DO 10 I=1,M
      ISUB=SUB(I)
      DO 10 J=1,N
10    B(I,J)=A(ISUB,J)
      DO 20 I=1,M
      DO 20 J=1,N
20    A(I,J)=B(I,J)
      RETURN
      END
PROGRAM  SORT   HAS    NO ERRORS
```

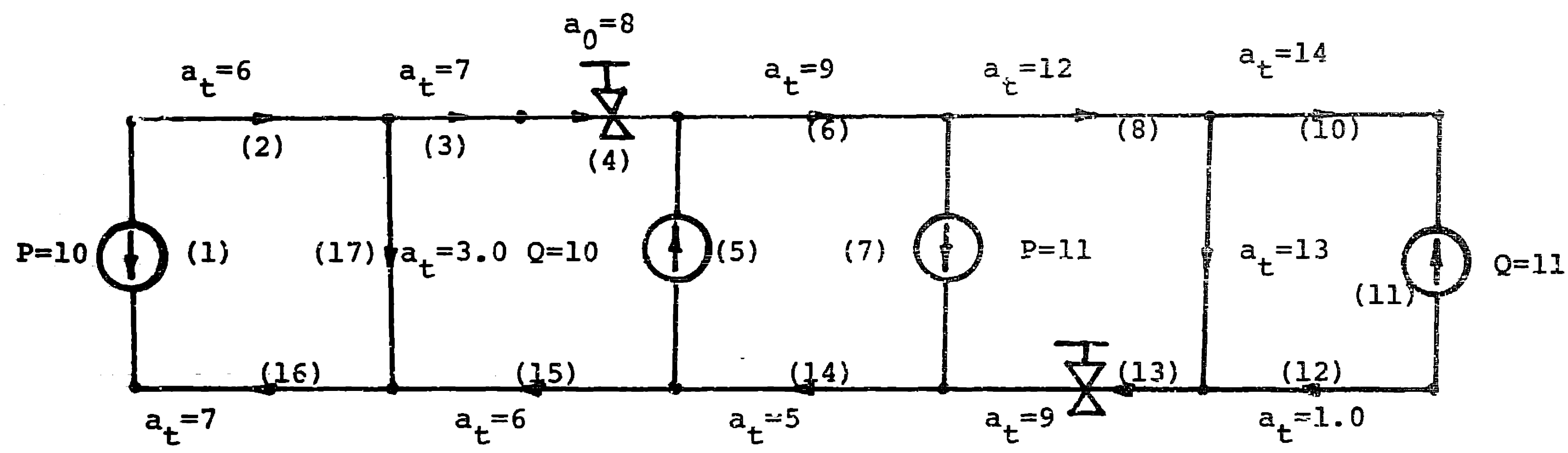
```
USER=GOODMAN 274 84255    JOINT COMPUTER FACILITY, MIT

C  PROGRAM:      ICSORT
C
C  ARGUMENTS:    A - MATRIX TO BE SORTED
C
C                M - NUMBER OF ROWS OF A
C                N - NUMBER OF COLUMNS OF A
C                SUB - VECTOR OF SUBSCRIPTS
C
C  DESCRIPTION: ICSORT (INTEGER COLUMN SORT) SORTS
C                THE COLUMNS OF AN INTEGER MATRIX,
C                A, ACCORDING TO THE ORDER DICTATED
C                BY THE VECTOR SUB.
C
C
C
      SUBROUTINE ICSORT(A,M,N,SUB)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 A(30,30),B(30,30)
      INTEGER*2 SUB(N)
      DO 10 J=1,N
      JSUB=SUB(J)
      DO 10 I=1,M
10    B(I,J)=A(I,JSUB)
      DO 20 I=1,M
      DO 20 J=1,N
20    A(I,J)=B(I,J)
      RETURN
      END
PROGRAM  ICSORT HAS    NO ERRORS


O OPERATING SYSTEM VERSION 2 REVISION  222  6/13/76 GENERATED  12/17/76  03:26.


HANDLING CHARGE    $  .35 / JOB        .35
LINES PRINTED PR1 $ 1.25 / K LN        .89
CARDS READ         $ 1.50 / K CD      1.03
PLOTTER VECTORS    $  .25 / 1000       .00
MODEL 70 SECONDS  $25.00 / HOUR        .34
MODEL 80 SECONDS  $35.00 / HOUR        .00
                    TOTAL CHARGE $   2.61


 274 84255  LOGGED OUT 05/19/17 23:45.  $   6.54 LEFT AFTER    16 LOGINS.
```



The network above is solved in the program run which follows.

```
// JOB ANY 00080                                           GOODMAN     84255
// DUP
XB  >
*PN CHLDWAT
*EL SSPMAT
*EL BINARY
*BC OFBA
*LC 0064
*LAST
CHLDWAT   SSPMAT   BINARY7   RTL
 EOF
 EOF
 EOF
THURSDAY  05/19/17  23:37:05  @


 PROGRAM LABELS:
   1C00 *MAIN*   3DC6 INIT     6972 F        6D80 PRESS    6DE4 OUTPUT   6
   7508 ASSIGN   755C ERROR1   75DC ERROR2   766E ERROR3   76F8 ERROR4   7
   8F5C ICSORT   97CC MINV     B26C ALOG     B364 AEXP     B364 EXP      E
   B5DC ABS      B834 EXIT     B8DA


 COMMON-BLOCKS:
   F044 //       EF90 BLK1     EFCC BLK2


 UNDEFINED SUBROUTINES:
 NONE


 TRANSFER ADDRESS 1C00
PROGRAM ENDS AT: X'BA7A' AND REQUIRED 006A DISK READS;0068 DISK WRITES@
FIND REFS=0663 RECALL BUFS= 0066@
EXECUTION BEGINS: @
```

PIPE NO.	PRESSURE	FLOW RATE
8	-162.4736	-4.4323
7	11.0000	10.2744
2	-5.8342	-0.9841
11	-1346.8977	11.0000
5	-318.3394	10.0000
9	350.2810	6.5677
10	-930.1758	-11.0000
12	-66.4411	-11.0000
13	-176.8076	-4.4323
14	109.7641	5.8421
15	-72.6421	-4.1579
16	-6.8066	-0.9841
17	22.6407	3.1738
1	10.0000	0.9841
3	-84.7491	-4.1579
4	-138.3076	-4.1579

6 197.5754 5.8421

// END

JCF VIO OPERATING SYSTEM VERSION 2 REVISION 222 6/13/76 GENERATED 12/17

JOB HANDLING CHARGE	$ .35 / JOB	.35
51 LINES PRINTED PR1	$ 1.25 / K LN	.06
28 CARDS READ	$ 1.50 / K CD	.04
00 PLOTTER VECTORS	$ .25 / 1000	.00
35 MODEL 70 SECONDS	$25.00 / HOUR	.24
00 MODEL 80 SECONDS	$35.00 / HOUR	.00
	TOTAL CHARGE $	.69

GOODMAN 274 84255 LOGGED OUT 05/19/17 23:37. $ 9.15 LEFT AFTER 15 LO