THE CONSIST ROUTINE
A COMPUTER PROGRAMME FOR SOLVING SIMULTANEOUS EQUATIONS

By V. C. SABHERWAL
Indian Statistical Institute

SUMMARY. This paper presents a routine for solving a system of equations satisfying certain conditions. This routine is written in Fortran II for IBM-1620 with 40 K core storage memory and card input-output system. With slight changes, this routine can be made to work on CDC-3600 and IBM-1620 with on-line printer.

1. Introduction

This computer routine has been written to provide specific solutions to a system of simultaneous equations of the type

$$A\,X = B$$

or

$$\begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1m} \\ \cdot & & \cdot & & \cdot \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{im} \\ \cdot & & \cdot & & \cdot \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ \cdot \\ x_j \\ \cdot \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \cdot \\ b_i \\ \cdot \\ b_m \end{bmatrix}$$

Here X is the unknown vector to be solved for. The square matrix A and the right-hand-side vector B are known. This routine also gives a complete set of row-by-row balances; i.e. the product terms $a_{ij}x_j$ and the row sums $\sum_{j=1}^{m} a_{ij}x_j$. Throughout the discussion of this routine, the elements of the matrix A will have two subscripts. The subscript i will always stand for the row, and the subscript j will stand for the column of the matrix to which the element belongs.

The method used in this routine is an iterative procedure, and is capable of solving those systems which conform to the Leontief conditions for input-output models. A sufficient condition for the convergence of the method is discussed in Section 3. Without making use of magnetic tapes, this routine for the IBM 1620 can handle problems containing less than 100 variables and also less than 800 non-zero coefficients in the matrix.

2. Methodology

This routine solves the problem in a number of iterations. (The factors influencing the number of iterations in a particular problem will be discussed later.) The vector X is called a solution vector and the corresponding vector Y, defined as $Y = B - AX$, is called the error vector. We start the first iteration with the solution $X = 0$, and hence the error vector $Y = B$. Each iteration consists of m (equal to the number of variables) steps, and all steps are repeated at every iteration. The j-th step of any iteration is as follows :

(a) Calculate Δx_j by the relations :

$$\Delta x_j = \frac{y_j^{j-1}}{a_{jj}}$$

where y_j^{j-1} is the j-th component of the error vector Y at the $(j-1)$-th step.

(b) Calculate the solution vector X at the j-th step from the solution vector at the $(j-1)$-th step as :

$$X^j = X^{j-1} + \Delta X^j$$

where $\Delta X^j = (0 \ldots \Delta x_j \ldots 0)$ is a column vector.

(c) Calculate the error vector Y at the j-th step as :

$$Y^j = B - AX^j = Y^{j-1} - A\Delta X^j.$$

Thus, in each step, we modify the solution vector and reduce to zero the j-th component of the error vector Y.

After completing all the m steps in an iteration, we test the error vector against the tolerance limit set *a priori*. The iterations are halted when

$$|Y| \leqslant \text{tolerance}.$$

At each iteration, we modify only those variables i for which $|y_i|$ exceeds the tolerance limit.

In this iterative procedure, note that we make use of the individual coefficients a_{ij} sequentially, following the order of the column vectors. These coefficients constitute the vast bulk of what is stored in memory. In designing a programme for a magnetic tape machine, this sequential access feature would be particularly advantageous.

3. A sufficient condition for convergence

To achieve neatness in the proof and with no loss in generality, let us assume that

$$a_{jj} = 1 \qquad \text{for } j = 1, 2, \ldots, m.$$

With this assumption, a sufficient, but not necessary, condition for convergence is

$$\sum_{i \neq j} |a_{ij}| < 1.$$

Convergence of the routine will be proved if :

(i) sum of absolute errors goes down at each step or maintains the same value, and

(ii) sum of absolute errors can be made to go below any predetermined level in a finite number of steps.

Proof: In Section 2, we have seen that

$$Y^j = B - AX^j$$

$$Y^{j-1} = B - AX^{j-1}.$$

From these two relations, together with $a_{jj} = 1$, we can write

$$Y^j = Y^{j-1} - A(X^j - X^{j-1}) \qquad \ldots (1)$$

and

$$X^j - X^{j-1} = \Delta X_j = \begin{bmatrix} 0 \\ y_j^{j-1} \\ 0 \end{bmatrix} \qquad \ldots (2)$$

From (1) and (2) we have :

$$|y_j^j| = |y_j^{j-1}| - |y_j^{j-1}| = 0 \qquad \ldots (3)$$

and

$$y_i^j = y_i^{j-1} - a_{ij}\, y_j^{j-1} \qquad \ldots (i \neq j)$$

$$\therefore \quad |y_i^j| = |y_i^{j-1} - a_{ij}\, y_j^{j-1}| \qquad \ldots (1 \neq j)$$

$$\therefore \quad |y_i^j| \leqslant |y_i^{j-1}| + |a_{ij}|\, |y_j^{j-1}| \ \ldots (i \neq j). \qquad \ldots (4)$$

Summing (4) over all $i \neq j$, we have :

$$\sum_{i \neq j} |y_i^j| \leqslant \sum_{i \neq j} |y_i^{j-1}| + |y_j^{j-1}| \sum_{i \neq j} |a_{ij}|. \qquad \ldots (5)$$

Addition of (3) and (5) yields :

$$\sum_{i=1}^{n} |y_i^j| \leqslant \left[\sum_{i=1}^{n} |y_i^{j-1}| \right] - |y_j^{j-1}| \left[1 - \sum_{i \neq j} |a_{ij}| \right]. \qquad \ldots (6)$$

If

$$\sum_{i \neq j} |a_{ij}| < 1.$$

then

$$\sum_{i=1}^{n} |y_i^j| \leqslant \sum_{i=1}^{n} |y_i^{j-1}|. \qquad \ldots (7)$$

Relation (7) proves statement (i). Note further that the equality sign holds in (7) if and only if $y_j^{j-1} = 0$. In this case, the routine automatically bypasses the j-th step.

To prove statement (ii), let us assume that at every step j we reduce the maximum of the error components to zero rather than proceeding sequentially from y_1 to y_2 to ... y_m. Further define S and k so that :

$$S = \sum_{i=1}^{m} |y_i^{j-1}|$$

and

$$\max_i |y_i^{j-1}| = |y_k^{j-1}|$$

$$\therefore \quad |y_k^{j-1}| \geqslant S/m. \qquad \ldots (8)$$

From relation (6) we can write :

$$\sum_{i=1}^{n} |y_i^j| - S \leqslant -|y_k^{j-1}| \left\{ 1 - \sum_{i \neq k} |a_{ik}| \right\}$$

or

$$S - \sum_{i=1}^{n} |y_i^j| \geqslant |y_k^{j-1}| \left\{ 1 - \sum_{i \neq k} |a_{ik}| \right\}. \qquad \ldots (9)$$

From (8) and (9) :

$$S - \sum_{i=1}^{m} |y_i| > \frac{S}{m}\left\{1 - \sum_{i \neq k} |a_{ik}|\right\}$$

$$S\left[1 - \frac{1}{m}\left\{1 - \sum_{i \neq k} |a_{ik}|\right\}\right] > \sum_{i=1}^{m} |y_i| .$$

Thus, at each step, the sum of absolute errors is multiplied by a factor not exceeding $\left[1 - \frac{1}{m}\left\{1 - \sum_{i \neq k} |a_{ik}|\right\}\right]$, a quantity less than one. This leads to the conclusion that the sum of absolute errors is decreasing at least as rapidly as a geometric series and hence can be made to go below any predetermined positive level in a finite number of steps.

4. Comments on convergence

Two problems were solved on an IBM 1620 (punched card input-output system; 40,000 characters of magnetic core memory). Both problems satisfied the following conditions :

$$\text{tolerance} = .0001$$
$$b_i \leqslant 10.000$$

The first problem satisfied the Leontief conditions and

$$.001 \leqslant |a_{ij}| \leqslant 1.000.$$

It contained 29 equations, 400 non-zero elements, and was solved in 7 iterations. The second was a 35 equation problem (320 non-zero elements). This did not satisfy the Leontief conditions, but

$$.001 \leqslant |a_{ij}| \leqslant 4.000$$

was satisfied. The computer solved it in 11 iterations. The two problems took 8 minutes and 12 minutes respectively.

One factor believed to influence the speed of the routine is the triangularity of the matrix A. It is recommended that the matrix A be arranged so as to get the maximum number of zeros above and to the right of the main diagonal. It is to be noted that although triangularity of the matrix is a desirable condition, it is not essential for the convergence and operation of the routine.

5. Sample problem

For checking out this programme, the following sample problem may be tried.

$$.8x_1 - .4x_3 = 2.0$$
$$-.5x_1 + 1.0x_2 = 2.5$$
$$-.5x_2 + 1.0x_3 = 2.5$$

A complete listing of the input data cards and output is given at the end. The input data cards are to be prepared as described below.

6. Data preparation

The data must be presented to the machine in the following order :

(i) Master card

(ii) Right-hand-side constants

(iii) Matrix coefficients

(iv) 999-card

(i) *Master card.* This card gives general information about the problem to the computer. The information is to be punched on one card in the following form :

(a) Punch the number of rows (right justified) in the matrix A in columns 1-3 of the card.

(b) Punch the number of non-zero elements (right justified) in the matrix A in columns 4-7 of the card.

(c) Punch numerical tolerance in columns 8-13 of the card. The decimal point in the numerical tolerance should be punched in column 8. The numerical tolerance may, therefore, be anywhere between .00001 and .99999. The iterations will end when the absolute value of the error in each equation is below the tolerance. That is, $|Y| = |B - AX| \leqslant$ tolerance.

In this sample problem, there are three equations and 6 non-zero coefficients. The numerical tolerance was put at .00100. Thus the first card of this problem is of the following form :

Col.	Col.	Col.
1–3	4–7	8–13
003	0006	.00100

(ii) *Right-hand-side constants.* All right-hand-side constants, including constants with zero values, are listed. The listing should be one element per card in the format given in Table 1.

TABLE 1. FORMAT FOR RHS VECTOR

description of field	solution vector	sign	leading digits (right justified)	decimal point	fractional portion	row number
col. no. of the card	5 6 7	16	17-20	21	22-27	31-33
1st card	0 . 0		0002	.	000000	001
2nd card	0 . 0		0002	.	500000	002
3rd card	0 . 0		0002	.	500000	003

A blank in all columns, except the 10th column, will be interpreted as zero. In column 10, a blank will be interpreted as a positive sign, and a minus as a negative sign.

All the right-hand-side cards should be arranged in ascending order of the row number.

(iii) *Matrix coefficients.* Only non-zero matrix coefficients are to be listed. There is a distinct advantage in setting up the routine in this way. If we were dealing with a 50×50 matrix, we would otherwise have to punch 2500 cards. In large consistency models, a majority of the entries are likely to be zero. In the 35-equation problem, discussed earlier, there were only 320 non-zero elements out of a possible $35 \times 35 = 1225$ entries. Thus this routine is easier to handle as we need punch non-zero elements only, and it is quicker as only the non-zero elements are operated upon.

The elements are to be listed one per card according to the format given in Table 2, which also gives a listing of all the elements of the sample problem.

TABLE 2. FORMAT FOR MATRIX COEFFICIENTS

description of field	column number j	row number i	sign	leading digit (right justified)	decimal point	fractional portion
column numbers in the card	1-3	4-6	7	8-11	12	13-18
1st card	001	001		0000	.	800000
2nd card	001	002	—	0000	.	500000
3rd card	002	002		0001	.	000000
4th card	002	003	—	0000	.	500000
5th card	003	003		0001	.	000000
6th card	003	001	—	0000	.	400000

Blank column everywhere, except column number 7, will be interpreted as zero. In column 7, a blank will be interpreted as a positive sign, and a minus as a negative sign.

Cards for the matrix should be arranged columnwise ; i.e., all cards belonging to the first column vector should come first, then the cards belonging to the second column, and so on. In beginning a new column vector, the first coefficient listed *must* be the diagonal element. The remaining cards of the column vector may be in any convenient order.

If the cards are not arranged in the way mentioned above, the computer will come to the "Stop" mode as soon as it finds the first out-of-order card. The consol typewriter will type the column number and row number of the element punched in such a card.

(iv) The last card of the data deck must be a 000-card. In this card, the number 000 is to be punched in columns 1-3. This card functions within the programme to compare the number of cards actually read in with the number supposed to have been read.

7. Operations on the IBM 1620

Three out of four sense-switches have been incorporated in the routine.

Sense-switch 1 *on* will punch the results of all the intermediate iterations on to cards. If the results of the intermediate iterations are not required, sense-switch 1 is to be turned *off*.

Sense-switch 2 *on* will punch the results of the final iteration on to cards. If it is turned *off*, the results of the last iteration are typed on the console typewriter.

Sense-switch 3 *on* will give the row-by-row balances, i.e., the non-zero terms $a_{ij}\,x_j$ and the sums $\sum_{j=1}^{m} a_{ij}\,x_j$. If sense-switch is *off*, these terms will neither be computed nor punched out.

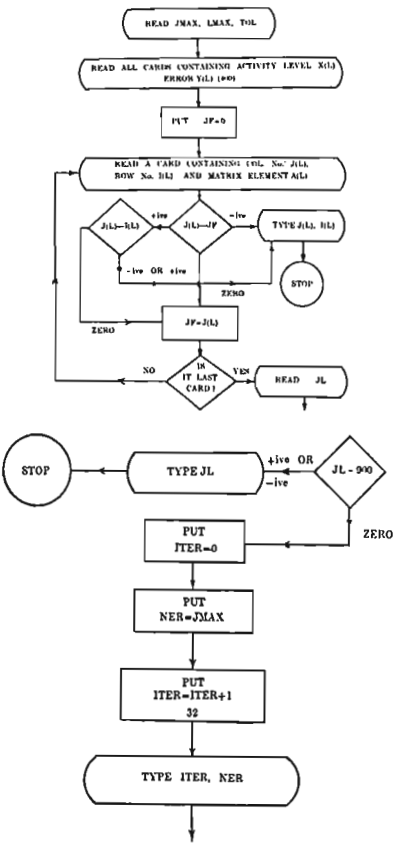
As soon as the data are fed into the computer, it will type 001 followed by a number equal to the number of rows in the matrix. This marks the completion of the first iteration. After completion of each subsequent iteration, the console typewriter will type the number of the iteration it has just completed and the number of errors at this iteration, i.e., the number of rows i for which $|y_i|$ is greater than the tolerance. The computer will continue the iterations until there are no more rows whose absolute errors lie above the tolerance level.

The results of different iterations are transmitted out according to the following format: the activity levels x_j are punched out in columns 1-12 of a card. The error terms y_i are punched in columns 16-27. (In the final iteration, these errors are all below the tolerance limit.) The identification of the activity is punched in columns 31-33. The number of the iteration the computer has just completed is punched in columns 37-39. All the above information is punched into one card. In any iteration, the number of such cards will be equal to the number of rows in the matrix.

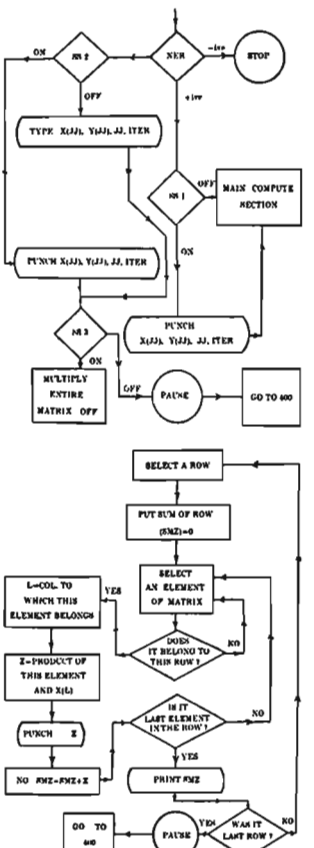
The results of the flow matrix are transmitted out in a different format. Each non-zero product $a_{ij}\,x_j$ is punched onto a different card. The column number of the product is punched in columns 1-3; the row number is punched in columns 4-6. The value of $a_{ij}\,x_j$ is punched in columns 7-18. After punching the product terms of row i, the sum of those product terms, $\sum_{j} a_{ij}\,x_j$, is punched onto the next card along with the row number.

Having completed each problem, the computer will come to rest in "Manual" position. This means the computer is ready to solve the next problem. Load the data cards for the next problem, if any, and press the "Start" key. On doing so, the IBM 1620, with card-input-output will say "Reader no feed". The computer will start reading the data cards, if "Reader Start" key is pressed on IBM cardreader.

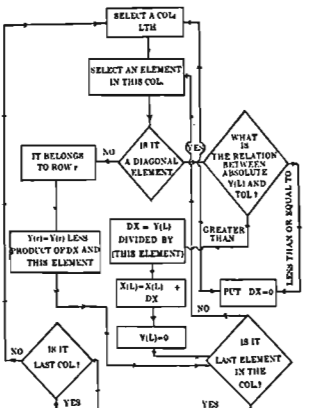
8. CONSIST FLOW DIAGRAM

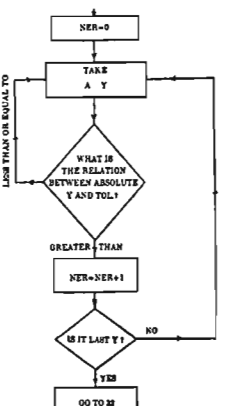


CONSIST FLOW DIAGRAM



CONSIST FLOW DIAGRAM





9. CONSIST SOURCE ROUTINE (FORTRAN II) FOR IBM 1620 WITH CARD-INPUT-OUTPUT SYSTEM

```
C  C     CONSIST 1C
       DIMENSION X(100), Y(100), J(800), I(800), A(800)
     1 FORMAT (I3, I4, F6.5)
     2 FORMAT (F12.6, 3X, F12.6, 3X, I3, 3X, I3)
     3 FORMAT (I3, I3, F12.6)
     4 FORMAT (3 HSUM, I3, F12.6)
   400 READ 1, J MAX, L MAX, TOL
       DO 5 JJ = 1, J MAX
     5 READ 2, X(JJ), Y(JJ)
C      READ L-ARRAY, CHECK SEQUENCE
       JF = 0.
       DO 9 L = 1, L MAX
       READ 3, J(L), I(L), A(L)
       IF (J(L)-JF)6, 9, 7
     6 TYPE 3, J(L), I(L)
       STOP
     7 IF (J(L)-I(L))6, 9, 6
     9 JF = J(L)
C      CHECK ON TOTAL NUMBER OF DATA CARDS
       READ 3, JL
       IF (JL-900)10, 30, 10
    10 TYPE 3, JL
       STOP
C      ITERATION OUTPUT, BUT BYPASS VIA SSION
    30 ITER = 0
       NER = J MAX
    32 ITER = ITER+1
       TYPE 3, ITER, NER
       IF (NER) 33, 37, 35
    33 STOP
    35 IF (SENSE SWITCH 1) 34, 36
    34 DO 39 JJ = 1, J MAX
    39 PUNCH 2, X(JJ), Y(JJ), JJ, ITER
       GO TO 36
    37 DO 45 JJ = 1, J MAX
       IF (SENSE SWITCH 2)38, 31
```

CONSIST SOURCE ROUTINE (FORTRAN II) FOR IBM 1620 WITH CARD INPUT-OUTPUT SYSTEM

```
      31  TYPE 2, X(JJ), Y(JJ), JJ, ITER
          GO TO 45
      38  PUNCH 2, X(JJ), Y(JJ), JJ, ITER
      45  CONTINUE
C         MULTIPLY ENTIRE MATRIX VIA SS3 ON
          IF (SENSE SWITCH 3) 40, 47
      40  DO 41 II = 1, J MAX
          SMZ = 0.
          DO 46L = 1, L MAX
          JL = J(L)
          IL = I(L)
          IF (IL-II)46, 44, 46
      44  Z = X(JL)*A(L)
          PUNCH 3, JL, IL, Z
          SMZ = SMZ+Z
      46  CONTINUE
      41  PUNCH 4, II, SMZ
      47  PAUSE
          GO TO 400
C         MAIN COMPUTE SECTION
      36  DO 70 L = 1, L MAX
          IF (I(L)-J(L))55, 51, 55
C         BRANCH FOR DIAGONAL ELEMENTS
      51  IL = I(L)
          JL = IL
          IF (Y(IL))300, 300, 301
     300  ABY = -Y(JL)
          GO TO 302
     301  ABY = Y(JL)
     302  IF (ABY-TOL)52, 52, 53
      52  DX = 0.
          GO TO 70
      53  DX = Y(IL)/A(L)
          X(JL) = X(JL)+DX
          Y(JL) = 0.
          GO TO 70
```

CONSIST SOURCE ROUTINE (FORTRAN II) FOR IBM 1620 WITH CARD INPUT-OUTPUT SYSTEM

```
C       BRANCH FOR OFF-DIAGONAL ELEMENTS
    55  IF (DX)60, 70, 60
    60  IL = I(L)
        Y(IL) = Y(IL)−DX*A(L)
    70  CONTINUE
C       COUNT OF NER
        NER = 0
        DO 80 JJ = 1, J MAX
        IF (Y(JJ))71, 71, 72
    71  ABY = −Y(JJ)
        GO TO 73
    72  ABY = Y(JJ)
    73  IF (ABY−TOL)80, 80, 75
    75  NER = NER+1
    80  CONTINUE
        GO TO 32
        END
```

10. MACHINE INPUT

(i) *Master card*

30000·00100

(ii) *RHS coefficients with row designations*

2·0	001
2·5	002
2·5	003

(iii) *Matrix coefficients with column row designations*

001001 ·8
001002− ·5
002002 1·
002003− ·5
003003 1·
003001− ·4

(iv) 900-*Card*

11. MACHINE OUTPUT

(i) *Results of all iterations*

Sol. Vec.	Error Vector	Row no.	Iter. no.
·0000	2·000000	1	1
·0000	2·500000	2	1
·0000	2·500000	3	1
2·5000	1·750000	1	2
3·7500	0·000000	2	2
4·3750	0·000000	3	2
4·6875	·218750	1	3
4·843750	0·000000	2	3
4·921875	0·000000	3	3
4·960037	·027343	1	4
4·980468	0·000000	2	4
4·990234	0·000000	3	4
4·995117	·003417	1	5
4·997558	0·000000	2	5
4·998779	0·000000	3	5
4·999389	·000427	1	6
4·999694	0·000000	2	6
4·999847	0·000000	3	6

(ii) *Flow matrix output with column row designations*

1	1	3·999511
3	1	−1·999938
Sum	1	1·999572
1	2	−2·499694
2	2	4·999694
Sum	2	2·500000
2	3	−2·499847
3	3	4·999847
Sum	3	2·500000

ACKNOWLEDGEMENTS

The author is grateful to Professor Allan S. Manne and Dr. T. N. Srinivasan for the keen and helpful interest they have taken in the preparation of this paper.

Paper received : December, 1964.