UNITED STATES

DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

DIRECT SOLUTION ALGORITHM FOR THE TWO DIMENSIONAL GROUND-WATER FLOW MODEL

Open-File Report 79-202

UNITED STATES

DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

DIRECT SOLUTION ALGORITHM FOR THE TWO DIMENSIONAL GROUND-WATER FLOW MODEL

By S.P. Larson

Open-File Report 79-202

Reston, Va

June, 1978

TABLE OF CONTENTS

ILLUSTRATIONS

ABSTRACT

Alternating diagonal ordering of node points for a two-dimensional finite-difference model of ground-water flow can be used to produce a direct solution algorithm that is computationally more efficient than iterative methods for moderately sized grids. Comparisons with the strongly implicit procedure, line-succesive overrelaxation, and the iterative alternating direction implicit procedure indicate that a direct method using alternating diagonal ordering can be competitive for as many as 3,000 equations. A FORTRAN computer code is included that is compatible with the two-dimensional ground-water flow model developed by the U.S. Geological Survey. The performance characteristics, computer storage requirements, and input data requirements for the direct solution algorithm are also included.

INTRODUCTION

As the availability of large capacity, high speed computers increases, the utility of direct methods (Gaussian elimination) for solving the set of linear algebraic equations encountered in ground-water modeling also increases. Price and Coats (1974) analyzed the use of direct methods for solving matrix equations encountered in reservoir simulation problems. They argue that it is well known that the commonly used method for ordering equations (that is, numbering a finite-difference grid in the smallest dimension) is certainly not the most efficient one. They go on to discuss the advantages of various alternative methods for ordering equations, in particular, a method which they refer to as D4 or alternating diagonal ordering. Results indicate that for large grids, D4 ordering requires only one-fourth the computing time and one-third the storage of standard ordering for non-symmetric problems in two-dimensions.

D4 ORDERING

The purpose of D4 ordering is to construct a coefficient matrix such that during the elimination process, sparsity will be conserved. Sparsity refers to the relative number of non-zero elements in the matrix. Certain multiplications and divisions can be avoided if zero elements are encountered during elimination and thus, if the sparsity is maximized, the work required to complete the elimination can be minimized. Consider a 5-by-5 grid shown in figure 1 with the grid points numbered in D4 fashion. The coefficient matrix [A], resulting from finite-difference approximations for a two-dimensional ground-water flow model will have non-zero entries denoted by the X's in figure 2.

Note that the upper half of [A] is already in upper triangular form (no non-zero elements to the left of the main diagonal). Eliminating unknowns associated with equations in the upper half from the equations in the lower half, produces non-zero entries in the lower half of [A] shown by the circles in figure 2. Note that, 1) calculations are not required for zero entries during this elimination, and 2) the bandwidth of non-zero entries created in the lower half is such that elimination through the lower half requires less work than standard ordering. Although item 2) may not be obvious from figure 2, Price and Coats (1974) demonstrate that these characteristics can reduce the work (number of multiplications and divisions) required for elimination to almost $N^2/4$ for large square grids, where N is the number of equations. Standard ordering requires N^2 multiplications and divisions; thus D4 ordering may require only one-fourth as much work.

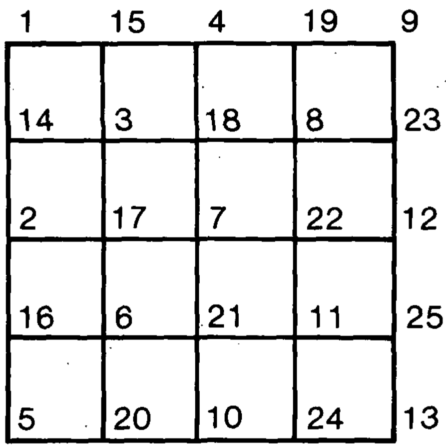

Figure 1.--D4 (alternating diagonal) ordering for a 5-by-5 grid.

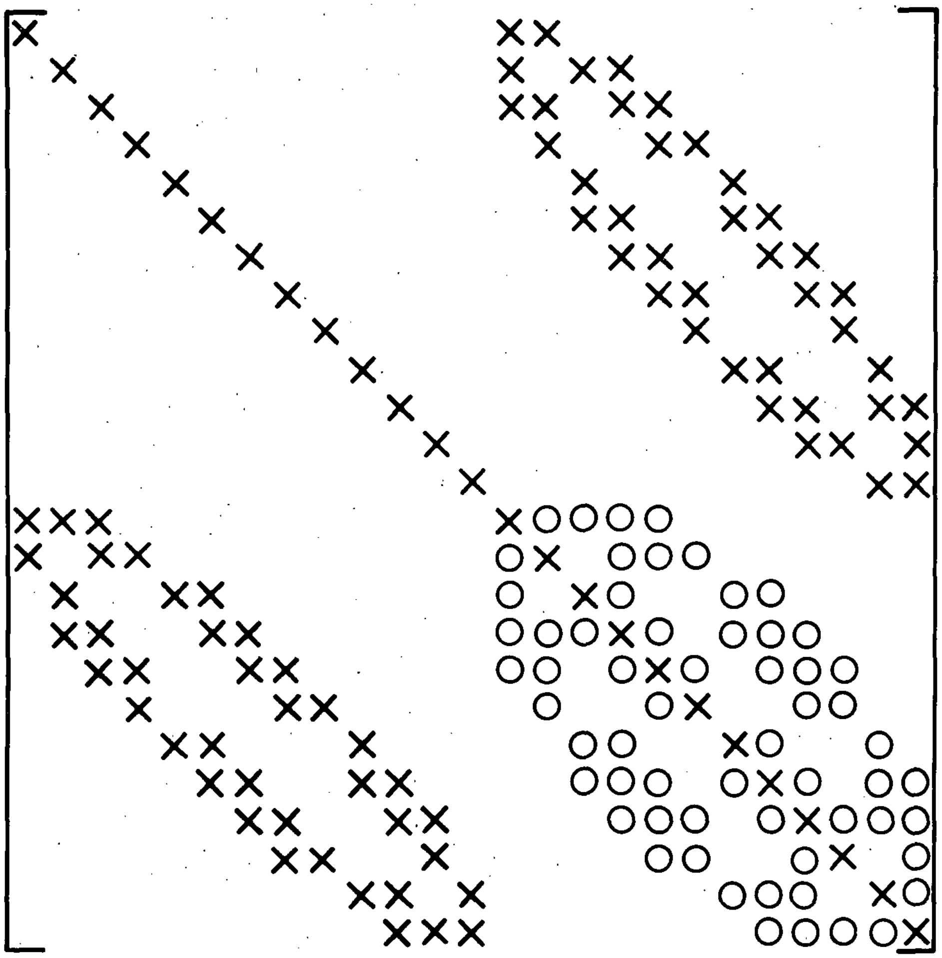

Figure 2.-- Structure of matrix [A] assuming D4 ordering. The X characters denote non-zero elements in the original matrix [A]. The 0 characters denote non-zero elements formed by eliminating the X characters from the equations in the lower half of the matrix.

Also, symmetric matrices require only one-half as much work as non-symmetric matrices (operations are necessary only to the right of the main diagonal). Thus, the work required using D4 ordering may approach $N^2/8$ or $IJ^3/8$ for large square grids, where I and J are the grid dimensions.

ESTIMATING WORK RATIOS

For direct solution methods, the bandwidth of the coefficient matrix is an important characteristic because the storage requirements are proportional to the bandwidth and work is proportional to the square of the bandwidth. The work required for elimination of a banded symmetric matrix, using standard ordering, is approximately $NJ^2/2$ or $IJ^3/2$ where J, the smallest grid dimension, is assumed to approximate the bandwidth of the matrix. If the reduction in work produced by D4 ordering can be estimated, the work ratio between D4 ordering and iterative methods can also be estimated for various grid sizes.

If $J<I$, the bandwidth for standard ordering is $J+1$ and the work for large I and J is, as mentioned above, approximately $IJ^3/2$. Therefore:

$$W_{D4} \simeq f_{D4} \frac{IJ^3}{2} \tag{1}$$

where f_{D4} is the work ratio of D4 compared to standard ordering. Figure 3 shows work ratios of D4 to standard ordering (f_{D4}) achieved using an IBM 370/155 computer for various grid sizes and grid elongations (ratios of J to I). The Gauss-Doolittle method of decomposition (Forsythe and Moler, 1967) was used for both D4 and standard ordering. Thus an estimate of work using D4 ordering can be obtained using figure 3 and equation 1.

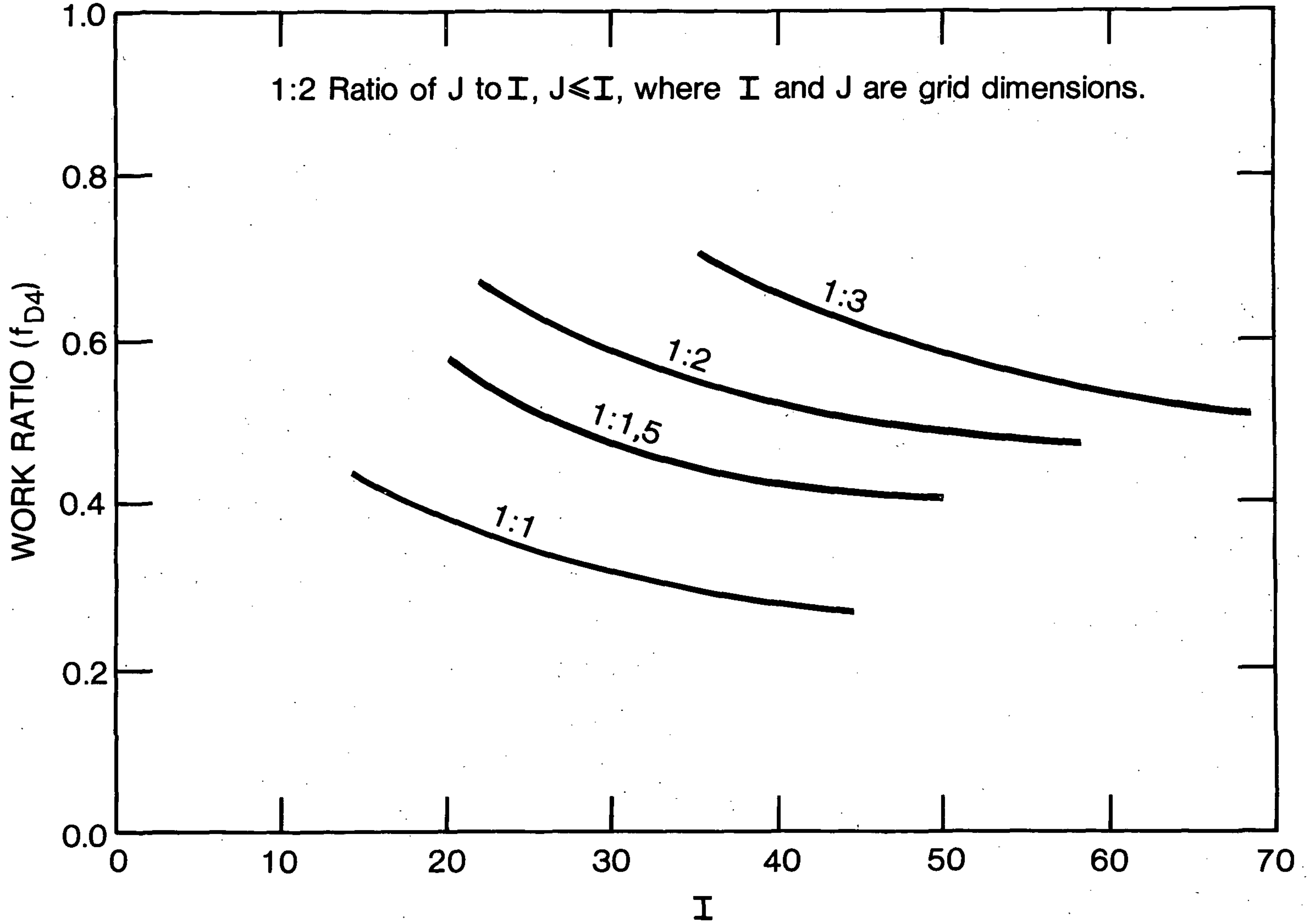

Figure 3.--Work ratio (f_{D4}) for various elongation ratios using D4 ordering.

For iterative solution methods, the work for each iteration is directly proportional to the number of equations and the total work required for a solution can be written:

$$W_{it} \simeq C_i N_i IJ \tag{2}$$

where C_i is the number of multiplications and divisons required per iteration, N_i is the number of iterations required for a solution, and IJ is the product of the grid dimensions which presumably approximates the number of unknowns for a given problem. The coefficient C_i is about 31 for SIP (strongly implicit procedure), 47 for IADI (iterative alternating direction implicit procedure) and 23 for LSØR (line-successive overrelaxation) as coded in the model for two-dimensional ground-water flow developed by Trescott and others (1976). Note that the grid dimensions of the two-dimensional ground-water flow model are not exactly equal to I and J as discussed herein. To simplify computations, the model grid includes a border of inactive node points. Thus the model grid dimensions must be reduced by 2 to obtain the values of I and J used in this discussion.

The relative work between the D4 method and the iterative methods can be estimated by combining equations 1 and 2 as:

$$\frac{W_{D4}}{W_{it}} \simeq \frac{0.5 f_{D4} J^2}{C_i N_i} \tag{3}$$

In developing a computer code that would be compatible with the two-dimensional ground-water flow model (Trescott and others, 1976), a small amount of overhead was required to calculate the coefficient matrix. To make a more accurate practical estimate of work ratios (W_{D4}/W_{it}), this overhead (approximately 20IJ multiplications) is included even though it becomes insignificant for large grids. The work ratio between D4 ordering and iterative methods can thus be approximated by:

$$\frac{W_{D4}}{W_{it}} \simeq \frac{0.5 f_{D4} J^2 + 20}{C_i N_i} \qquad (4)$$

Figure 4 depicts the quantity $W_{D4}N_i/W_{it}$ for various grid sizes (assuming I=J) for the three iterative methods included in the two-dimensional ground-water flow model. Equation 4 was used to construct the graph with values of f_{D4} obtained from figure 3 for a 1:1 elongation ratio. The quantity $W_{D4}N_i/W_{it}$ is the number of iterations that yield the same amount of work required by direct solution with D4 ordering. Thus if an iterative method requires more than $W_{D4}N_i/W_{it}$ iterations, the problem can be solved more efficiently using the D4 technique.

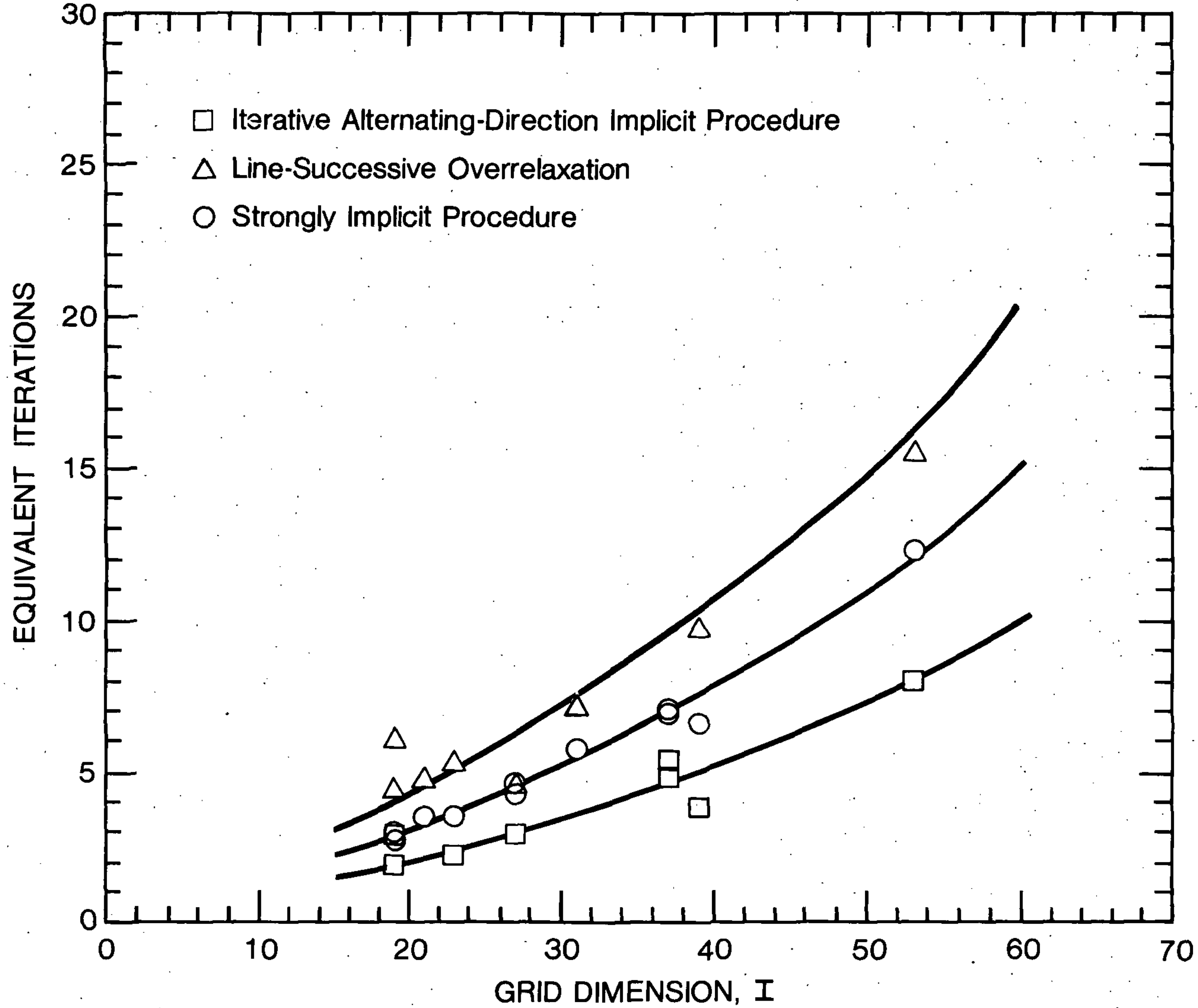

Figure 4.--Number of iterations that represent the same amount of work as direct solution; assuming D4 ordering.

It is also of interest to note that for a problem containing missing grid blocks (transmissivity equal to zero) or other irregularities in boundary geometry, the D4 technique may be more effective than equation 4 would predict. The reason is that missing grid blocks or irregular boundary geometry can result in a smaller bandwidth than that estimated from the grid dimensions. It is clear from equation 4 that if the bandwidth is reduced, the work required for the D4 technique may be significantly reduced because the work is directly proportional to the square of the bandwidth.

COMPUTER CODE

A FORTRAN computer code was developed to perform direct solution assuming D4 ordering. The code was constructed to be interchangeable with the SØLVE2 subroutine (LSØR) in the two-dimensional ground-water flow model (Trescott and others, 1976) and is listed in the appendix. Although the definition of some input data variables has changed, the only modification required to accommodate this subroutine into the program is to change one card in the main program. This card is also listed in the appendix. Before describing the changes in input data, a discussion of non-linear terms and uniform time steps is appropriate.

Non-linear Terms

For water-table aquifer systems; systems that include ground-water evapotranspiration; or combined water-table artesian simulations; the resulting equations are non-linear or are only piecewise linear. The term piecewise linear is meant to imply that the system is linear over certain ranges of head but not uniformly linear over the entire range. To analyze these problems effectively in the environment of a direct-solution scheme, linearization techniques such as Newton-Raphson iteration (Blair and Weinaug, 1969), or perturbation (J.V. Tracy, oral comm., 1977) can be used. Although these methods solve the problem in a mathematically pleasing fashion, a nonsymmetric coefficient matrix is produced, thus significantly reducing the utility of a direct-solution scheme. For most ground-water problems, a simple technique called extrapolation can give satisfactory results with a minimum of computational effort.

Extrapolation

The purpose of using a technique such as perturbation is to avoid decomposing the coefficient matrix more than once per time step, as would be required if the non-linear terms were updated iteratively. A very simple, yet effective, method for obtaining an estimate of the non-linear terms is to extrapolate the head using values calculated from preceding time steps (Von Rosenberg, 1969). Generally, extrapolation is made to the mid-point of the next time step, thus providing estimates of the average non-linear coefficients during that step. If the point of extrapolation is variable, the scheme could be written as

$$h^* = h_{k-1} + \theta(h_{k-1} - h_{k-2}) \qquad (5)$$

where h* is the estimated head to be used for calculating non-linear terms, h_{k-1} and h_{k-2} are heads at the k-1 and k-2 time levels, respectively, and θ is the extrapolation factor. If θ is set to zero, the scheme becomes one of explicit evaluation of non-linear terms at time level k-1. Although the method is simple in concept, it appears to be quite effective for many non-linear ground-water flow problems and yields an estimate of the solution to the non-linear problem in a single decomposition of the coefficient matrix.

Extrapolation may not eliminate all of the difficulties associated with non-linear terms, however, and so the computer code was structured to allow a sequence of "controlled" iterations during each time step. This takes the form of specifying a minimum number of iterations that must be completed during the step. Non-linear terms are evaluated using the head computed by the most recent iteration. A maximum number of iterations is also specified and the sequence is terminated if the maximum head change for an iteration is smaller than a specified tolerance. Termination of the sequence must be achieved within the maximum limit of iterations or the program will abort. However, by selecting an arbitrarily large closure tolerance, a minimum number of iterations can be guaranteed and the closure tolerance will be satisfied; thus the program will not abort. The use of iteration, although somewhat inefficient computationally, should allow the solution of many problems that cannot be solved using only extrapolation.

Uniform Time Steps

For linear problems (artesian simulations with no evapotranspiration), a direct-solution technique can be very effective for simulations with uniform time steps. For these problems, the coefficient matrix does not change from one time step to the next and therefore only a single decomposition of the matrix is required. Heads at subsequent time steps are determined by reformulating the right hand sides of the difference equations and back substituting. The computational work required to reformulate and back substitute can be substantially less than that of decomposition, thus solving for several uniform time steps can be accomplished much more efficiently than an equivalent number of non-uniform steps.

The computer code is designed to take advantage of this reduction in work automatically if the necessary conditions exist. The necessary conditions are: 1) artesian simulation, 2) no evapotranspiration, 3) no iteration specified (see variable LENGTH below), and 4) uniform time steps.

Changes to Input Data

Subsequent paragraphs describe changes in the definitions of some input data variables used in the two-dimensional ground-water flow model (Trescott and others, 1976). Complete descriptions of the input data cards can be found on pages 49-55 of that report.

In group II, card 2, columns 21-30, the variable ERR is used to define the error criterion for closure on the iteration sequence for non-linear problems. If the calculated head change for an iteration is smaller than this value at all nodes, iteration will stop. Reasonable values of this parameter are probably about 0.1 or 0.2 and are related to the amount of error in transmissivity, evapotranspiration coefficients, or leakage coefficients that is acceptable. A large value of ERR can be used to guarantee closure after a minimum number of iterations has been completed.

In group II, card 2, columns 71-80, the variable LENGTH is defined as the minimum number of iterations desired. Thus if at least 2 iterations (in addition to the first decomposition) are desired, code 2 for LENGTH. The maximum number of iterations desired is controlled by the parameter ITMAX (group I, card 4, columns 31-40). Set ITMAX to the maximum number of iterations desired. For some problems in which non-linearity is caused by the constraints on evapotraspiration coefficients or leakage coefficients in combined water-table artesian simulations, it may be desirable to iterate one or two times. If these two parameters (LENGTH and ITMAX) are set equal, and ERR is sufficiently large, LENGTH iterations will result. The purpose of this type of iteration is to insure that the water-level has not exceeded the allowable range for correct coefficient calculation during the time step. For example, evapotranspiration rate is limited to a maximum value if the water level is above land surface.

If the water level moves above land surface during a time step, the rate will be incorrect unless iteration is performed. However, this not be necessary for most problems and may only be significant for steady-state calculations. To avoid iteration, set LENGTH to zero.

In group II, card 3, columns 1-10, the variable HMAX is defined as a dampening factor similar to β' used in the SIP algorithmn. It can be used to control oscillations for some highly non-linear water-table problems. (See Trescott and others, 1976, pp. 26-29).

Recall that the computer code was constructed as a replacement for subroutine SØLVE2 (LSØR) and thus LSØR must be selected in group I, card 3, columns 26-30 to designate direct solution. If direct solution is selected, an additional data card is required prior to the group IV data. The card inputs the variable THETA used for extrapolation in water-table simulations. The format is F10.0 (columns 1-10) and a blank card is required for simulations in which direct solution is selected and THETA is not used (non-water-table simulations).

Additional arrays (AU, AL, IC, B, and IN) are required for direct solution and are dimensioned explicitly in the subroutine. (See Appendix).The required dimensions for AU, AL, IC, B, and IN are computed by the program and displayed on the program output. These variables and must be dimensioned at least as large as indicated on the output if the program is to run successfully. Array IN should always be dimensioned by at least DIML-2 by DIMW-2 (DIML and DIMW are the model grid dimensions). Initially, the other arrays can be dimensioned as follows, assuming N $\simeq$ DIML x DIMW, AU and IC should be N/2 by 5, AL should be N/2 by DIML-1, and B should be N. If these estimates differ significantly from the computed values, it may be appropriate to recompile using the computed dimensions.

Storage Requirements and Computation Time

Although storage requirements and computation time will depend entirely upon the type of computer system available, experience on an IBM 370/155 [1/] will be presented to provide some insight into expected values.

The core storage in thousand- byte units (1 byte = 8 bits, 32 bit words) can be approximated by:

$$C \simeq 87 + 0.034\ N^{1.23} \tag{6}$$

where N is the number of active nodes (unknowns). This assumes that all options have been selected and that the Y array (see Trescott and others, 1976, p. 38) and the additional arrays required for D4 ordering are dimensioned exactly as required. Thus, for 1000 unknowns, 254K bytes of core storage are required. That part of this total required by the additional arrays in D4 is approximately;

$$C_{D4} \simeq \frac{7N + 0.5NB}{256} \tag{7}$$

where B is one less than the smallest grid dimension (DIML-1 or DIMW-1). On modern computers, core storage is commonly available in quantities that allow serious consideration of problems involving as many as

---

1/ The use of brand name in this report is for identification purposes only and does not imply endorsement by the U.S. Geological Survey.

three thousand unknowns. As a practical matter, two-dimensional ground-water models seldom have more than 3,000 unknowns and therefore the D4 ordering technique should be an effective solution method.

An empirical relation for CPU (central processor) time in seconds, excluding data input, is:

$$t = (4.82 \times 10^{-5})\, N^{1.69} \qquad (8)$$

This is the time required to complete an iteration, or a non-uniform time step, if iteration is unnecessary.

Roundoff Error

Roundoff error may cause difficulties for some problems if the magnitude of the elements of the coefficient matrix are highly variable. The decomposition of the matrix as written in the computer code in the appendix is carried out in single precision arithmetic and computers such as the IBM 370/155 that have a standard word size of 32 bits (6 to 7 decimal digits) can be prone to roundoff error. Computers that have larger standard word sizes (such as the CDC 7600 with 60 bit words) seldom have roundoff error problems.

Errors in the mass balance computed by the ground-water model are indications of roundoff error. If the error is large (greater than about one percent), it may be necessary to 1) carry out the decomposition in double precision arithmetic, 2) iterate on the residual of the difference equations, 3) use some form of scaling the coefficient matrix, or 4) use a computer that has a larger standard word size. Iteration on the residual is accomplished merely by forcing iteration (LENGTH>0). Scaling the coefficient matrix requires modification of the computer code and was found to be somewhat ineffective on a test problem that exhibited roundoff error difficulties.

Utility

It is anticipated that the D4 method will be most useful in the solution of steady-state problems. For the iterative methods (SIP, ADI, and LSØR) solutions to steady-state problems generally require many iterations unless the initial estimates of aquifer head are close to the solution. This is uncommon, however, and thus the D4 method should be very effective.

For transient problems, the aquifer head at the old time level is normally very close to the values at the new (unknown) time level and iterative methods can be used to obtain a solution in a few iterations. Large time steps however, will probably result in a situation similar to steady-state problems in that many iterations may be required by iterative methods and the D4 method may be more effective. Also, as indicated previously, transient simulations of linear problems using many time steps of equal size may be accomplished very efficiently using the D4 method.

CONCLUSIONS

The size and speed of modern computers have increased utility of direct- solution techniques as applied to ground-water modeling problems. The D4 ordering scheme with Gauss-Doolittle decomposition is competitive with iterative methods, such as SIP, IADI and LSØR, for many problems. The problem of selecting iteration parameters, restrictions on coefficient variation, and slowly converging or possibly non-converging sequences of estimates are virtually eliminated if direct solution is used.

Work ratios between the D4 method and the iterative methods can be estimated and an evaluation of the utility of the D4 method can be made. On an IBM 370/155 computer, the two-dimensional ground-water model can be programmed to solve for 3,000 unknowns in the same amount of CPU time required for about 13 SIP iterations. Thus, direct solution assuming D4 ordering can be an effective solution algorithmn for a wide range of ground-water modeling problems.

REFERENCES CITED

Blair, P.M. and Weinaug, C.F., 1969, Solution of two-phase flow problems using implicit difference equations: Soc. Petrol. Eng. Jour. Dec., 1969, p. 417-424.

Forsythe, C. E., and Moler, C. B., 1967, Computer solution of linear algebraic systems: New Jersey, Prentice-Hall, 148 p.

Price, H. S., and Coats, K. H., 1974, Direct methods in reservoir simulation: Soc. Petrol. Eng. Jour., June 1974, p. 295-308.

Trescott, P. C., Pinder, G. F. and Larson, S. P., 1976, Finite-difference model for aquifer simulation in two dimensions with results of numerical experiments: U.S. Geol. Survey, Techniques of Water Resources Inv., book 7, Chap. C1, 116 p.

Von Rosenberg, D. U., 1969, Methods for the numerical solution of partial differential equations: New York, American Elsevier Publishing Company, Inc., 128 p.

A P P E N D I X

Changes to program code to use D4

1) Change card MAN1710 in the Main program to:

```
43), Y(L(20)),Y,(L(22)),Y(L(21)),Y(L(18))
```

Note that this is a continuation card and thus the first character (4) is in column 6.

2) Insert the subroutine listed on the following pages in place of SØLVE2.

```
      SUBROUTINE SOLVE2(PHI,D1,D2,D3,KEEP,PHE,STRT,T,S,QRE,WELL,TL,
     1 SL,D14,D5,D6,D7,DELX,D8,DELY,D9,TEST3,TR,TC,GRND,SY,TOP,RATE,M,
     2 RIVER,BOTTOM)
C     SPECIFICATIONS:
      REAL *8PHI,E,RHO,CL,CR,CA,CB,AREA,DXB,DYB
      REAL *4KEEP,M,KEEPN
      INTEGER R,P,PU,DIML,DIMW,CHK,WATER,CONVRT,EVAP,CHCK,PNCH,NUM,HEAD,
     1CONTR,LEAK,RECH,SIP,ADI
C
      DIMENSION PHI(1),                        KEEP(1), PHE(1), STRT(1),
     1T(1), S(1), QRE(1), WELL(1), TL(1), SL(1),
     2        DELX(1),              DELY(1),             TEST3(1), TR(1), TC(1),
     3GRND(1), SY(1), TOP(1), RATE(1), M(1), RIVER(1), BOTTOM(1)
      DIMENSION AU(500,5), AL(500,31), IC(500,5), IN(50,50), P(1000)
C
      COMMON /SARRAY/ VF4(11),CHK(15)
      COMMON /SPARAM/ WATER,CONVRT,EVAP,CHCK,PNCH,NUM,HEAD,CONTR,EROR,LE
     1AK,RECH,SIP,U,SS,TT,TMIN,ETDIST,QET,ERR,TMAX,CDLT,HMAX,YDIM,WIDTH,
     2NUMS,LSOR,ADI,DELT,SUM,SUMP,SUBS,STORE,TEST,ETQB,ETQD,FACTX,FACTY,
     3IERR,KOUNT,IFINAL,NUMT,KT,KP,NPER,KTH,ITMAX,LENGTH,NWEL,NW,DIML,DI
     4MW,JNO1,INO1,R,P,PU,IXX,JXX,IDK1,IDK2
      RETURN
C     ..................................................................
C
C*******************
      ENTRY ITER2
C*******************
      READ 530, THETA
      IM=DIML-2
      JM=DIMW-2
C*****COMPUTE EQUATION NUMBERS FOR D4 ORDERING
      NXP=IM+JM-1
      DO 10 I=1,IM
      DO 10 J=1,JM
      N=I+J*DIML+1
      PHE(N)=STRT(N)
   10 IN(I,J)=0
      K=0
C*****ORDER--LEFT TO RIGHT, BOTTOM TO TOP
      DO 20 I=1,NXP,2
      DO 20 J=1,JM
      IK=I-J+1
      IF (IK.LT.1) GO TO 20
      IF (IK.GT.IM) GO TO 20
      N=IK+J*DIML+1
      IF(T(N).LE.0..OR.S(N).LT.0.) GO TO 20
      K=K+1
      IN(IK,J)=K
   20 CONTINUE
      ICR=K+1
      DO 30 I=2,NXP,2
      DO 30 J=1,JM
      IK=I-J+1
      IF (IK.LT.1) GO TO 30
      IF (IK.GT.IM) GO TO 30
      N=IK+J*DIML+1
      IF(T(N).LE.0..OR.S(N).LT.0.) GO TO 30
      K=K+1
      IN(IK,J)=K
   30 CONTINUE
C*****COMPUTE BANDWIDTH AND DETERMINE CONNECTING EQUATION NUMBERS
```

```
      MNO=9999
      MXO=0
      DO 80 I=1,IM
      DO 80 J=1,JM
      IR=IN(I,J)
      IF (IR.EQ.0.OR.IR.GE.ICR) GO TO 80
      JU=1
C**   LEFT
      IF ((J-1).LT.1) GO TO 40
      IF (IN(I,J-1).EQ.0) GO TO 40
      JU=JU+1
      IC(IR,JU)=IN(I,J-1)
      MM=IN(I,J-1)-IR
      MXO=MAX0(MM,MXO)
      MNO=MIN0(MM,MNO)
C**   ABOVE
   40 IF ((I-1).LT.1) GO TO 50
      IF (IN(I-1,J).EQ.0) GO TO 50
      JU=JU+1
      IC(IR,JU)=IN(I-1,J)
      MM=IN(I-1,J)-IR
      MNO=MIN0(MM,MNO)
      MXO=MAX0(MM,MXO)
C**   BELOW
   50 IF ((I+1).GT.IM) GO TO 60
      IF (IN(I+1,J).EQ.0) GO TO 60
      JU=JU+1
      IC(IR,JU)=IN(I+1,J)
      MM=IN(I+1,J)-IR
      MXO=MAX0(MM,MXO)
      MNO=MIN0(MM,MNO)
C**   RIGHT
   60 IF ((J+1).GT.JM) GO TO 70
      IF (IN(I,J+1).EQ.0) GO TO 70
      JU=JU+1
      IC(IR,JU)=IN(I,J+1)
      MM=IN(I,J+1)-IR
      MXO=MAX0(MM,MXO)
      MNO=MIN0(MM,MNO)
   70 IC(IR,1)=JU
   80 CONTINUE
      IB=MXO-MNO+2
      NEQ=K
      ICR1=ICR-1
      IB1=IB-1
      LH1=NEQ-ICR1
      LH=NEQ-ICR
      WRITE (P,510) HMAX,LENGTH,ITMAX,THETA
      WRITE (P,520) ICR1,LH1,IB1,ICR1,NEQ,IM,JM
      RETURN
C*******************
      ENTRY NEWITB
C*******************
      KOUNT=0
      ITYPE=0
      IF (CDLT.EQ.1..AND.KT.GT.1.AND.LENGTH.EQ.0.AND.EVAP.NE.CHK(6)) ITY
     1PE=1
      IF (WATER.NE.CHK(2)) GO TO 100
      ITYPE=2
      DO 90 I=1,IM
      DO 90 J=1,JM
```

```
      N=I+J*DIML+1                                                      D4 1180
      IF (T(N).LE.0..OR.S(N).LT.0.) GO TO 90                            D4 1190
      DELTAH=(PHI(N)-PHE(N))*CDLT*THETA                                 D4 1200
      DELMAX=0.1*(PHI(N)-BOTTOM(N))                                     D4 1210
      IF (ABS(DELTAH).GT.DELMAX) DELTAH=DELTAH*DELMAX/ABS(DELTAH)       D4 1220
      PHI(N)=PHI(N)+DELTAH                                              D4 1230
   90 CONTINUE                                                          D4 1240
      CALL TRANS                                                        D4 1250
  100 BIGI=0.                                                           D4 1260
C** LOAD MATRIX A AND VECTOR B FOR D4                                   D4 1270
      IF (ITYPE.EQ.1) GO TO 130                                         D4 1280
      DO 110 I=1,ICR1                                                   D4 1290
      DO 110 J=1,5                                                      D4 1300
  110 AU(I,J)=0.                                                        D4 1310
      DO 120 I=1,LH1                                                    D4 1320
      DO 120 J=1,IB1                                                    D4 1330
  120 AL(I,J)=0.                                                        D4 1340
  130 DO 140 I=1,NEQ                                                    D4 1350
  140 B(I)=0.                                                           D4 1360
      DO 310 I=1,IM                                                     D4 1370
      DO 310 J=1,JM                                                     D4 1380
      IF (IN(I,J).EQ.0) GO TO 310                                       D4 1390
      IR=IN(I,J)                                                        D4 1400
      N=I+1+DIML*J                                                      D4 1410
      NA=N-1                                                            D4 1420
      NB=N+1                                                            D4 1430
      NL=N-DIML                                                         D4 1440
      NR=N+DIML                                                         D4 1450
      DXR=DELX(J+1)                                                     D4 1460
      DYB=DELY(I+1)                                                     D4 1470
      STRTN=STRT(N)                                                     D4 1480
      KEEPN=KEEP(N)                                                     D4 1490
      PHEN=PHI(N)                                                       D4 1500
      IF (ITYPE.EQ.1) PHEN=PHE(N)                                       D4 1510
C     ..................................................................D4 1520
C                                                                       D4 1530
C     ---COMPUTE COEFFICIENTS---                                        D4 1540
      IF (EVAP.NE.CHK(6)) GO TO 160                                     D4 1550
C                                                                       D4 1560
C     ---COMPUTE EXPLICIT AND IMPLICIT PARTS OF ET RATE---              D4 1570
      GRNDN=GRND(N)                                                     D4 1575
      ETQB=0.                                                           D4 1580
      ETQD=0.0                                                          D4 1590
      IF (PHEN.LE.GRNDN-ETDIST) GO TO 160                               D4 1600
      IF (PHEN.GT.GRNDN) GO TO 150                                      D4 1610
      ETQB=QET/ETDIST                                                   D4 1620
      ETQD=ETQB*(ETDIST-GRNDN)                                          D4 1630
      GO TO 160                                                         D4 1640
  150 ETQD=QET                                                          D4 1650
C                                                                       D4 1660
C     ---COMPUTE STORAGE TERM---                                        D4 1670
  160 IF (CONVRT.EQ.CHK(7)) GO TO 170                                   D4 1680
      RHO=S(N)/DELT                                                     D4 1690
      IF (WATER.EQ.CHK(2)) RHO=SY(N)/DELT                               D4 1700
      GO TO 240                                                         D4 1710
C                                                                       D4 1720
C     ---COMPUTE STORAGE COEFFICIENT FOR CONVERSION PROBLEM---          D4 1730
  170 SUBS=0.0                                                          D4 1740
      TOPN=TOP(N)                                                       D4 1750
      IF (KEEPN.GE.TOPN.AND.PHEN.GE.TOPN) GO TO 210                     D4 1760
      IF (KEEPN.LT.TOPN.AND.PHEN.LT.TOPN) GO TO 200                     D4 1770
```

```
      IF (KEEPN-PHEN) 180,190,190
  180 SUBS=(SY(N)-S(N))/DELT*(KEEPN-TOPN)
      GO TO 210
  190 SUBS=(S(N)-SY(N))/DELT*(KEEPN-TOPN)
  200 RHO=SY(N)/DELT
      GO TO 220
  210 RHO=S(N)/DELT
  220 IF (LEAK.NE.CHK(9)) GO TO 240
C
C     ---COMPUTE NET LEAKAGE TERM FOR CONVERSION SIMULATION---
      IF (RATE(N).EQ.0..OR.M(N).EQ.0.) GO TO 240
      HED1=AMAX1(STRTN,TOPN)
      U=1.
      HED2=0.
      IF (PHEN.GE.TOPN) GO TO 230
      HED2=TOPN
      U=0.
  230 SL(N)=RATE(N)/M(N)*(RIVER(N)-HED1)+TL(N)*(HED1-HED2-STRTN)
  240 CONTINUE
C
      AREA=DXB*DYB
      E=(RHO+TL(N)*U+ETQB)*AREA
C*****LOAD COEFFICIENTS INTO AU AND AL
      CL=(TR(NL))*DYB
      CR=(TR(N))*DYB
      CA=(TC(NA))*DXB
      CB=(TC(N))*DXB
      IF (ITYPE.EQ.1) GO TO 300
      IF (IR.GE.ICR) GO TO 290
      JU=1
      IF ((J-1).LT.1) GO TO 250
      IF (IN(I,J-1).EQ.0) GO TO 250
      JU=JU+1
      AL(IR,JU)=-CL
  250 IF ((I-1).LT.1) GO TO 260
      IF (IN(I-1,J).EQ.0) GO TO 260
      JU=JU+1
      AL(IR,JU)=-CA
  260 IF ((I+1).GT.IM) GO TO 270
      IF (IN(I+1,J).EQ.0) GO TO 270
      JU=JU+1
      AL(IR,JU)=-CB
  270 IF ((J+1).GT.JM) GO TO 280
      IF (IN(I,J+1).EQ.0) GO TO 280
      JU=JU+1
      AL(IR,JU)=-CR
  280 E=E+CA+CB+CL+CR
      AL(IR,1)=E
      B(IR)=(RHO*KEEPN+SL(N)+GRE(N)+WELL(N)-ETQB+SUBS+TL(N)*STRTN)*AREA+
     1CA*PHI(NA)+CB*PHI(NB)+CL*PHI(NL)+CR*PHI(NR)-E*PHI(N)
      IF(T(N).GT.0.) GO TO 310
      AL(IR,1)=1.
      B(IR)=0.
      GO TO 310
  290 IRR=IR-ICR1
      E=E+CA+CB+CL+CR
      AL(IRR,1)=E
      B(IR)=(RHO*KEEPN+SL(N)+GRE(N)+WELL(N)-ETQB+SUBS+TL(N)*STRTN)*AREA+
     1CA*PHI(NA)+CB*PHI(NB)+CL*PHI(NL)+CR*PHI(NR)-E*PHI(N)
      IF(T(N).GT.0.) GO TO 310
      AL(IRR,1)=1.
```

```
      B(IR)=0.
      GO TO 310
  300 B(IR)=(RHO*KEEPN+SL(N)+GRE(N)+WELL(N)-ETQD+SUBS+TL(N)*STRTN)*AREA+
     1CA*PHI(NA)+CB*PHI(NB)+CL*PHI(NL)+CR*PHI(NR)-(E+CR+CL+CA+CB)*PHI(N)
      IF(T(N).GT.0.) GO TO 310
      B(IR)=0.
  310 CONTINUE
      IF (ITYPE.EQ.1) GO TO 380
C*****ELIMINATE TO FILL AL
      DO 340 I=1,ICR1
      JJ=IC(I,1)
      C1=1./AU(I,1)
      DO 330 J=2,JJ
      LR=IC(I,J)
      L=LR-ICR1
      C=AU(I,J)*C1
      DO 320 K=J,JJ
      KL=IC(I,K)-LR+1
      AL(L,KL)=AL(L,KL)-C*AU(I,K)
  320 CONTINUE
      AL(I,J)=C
  330 CONTINUE
  340 CONTINUE
C*****ELIMINATE AL
      DO 370 I=1,LH
      IR=I+ICR1
      L=I
      C1=1./AL(I,1)
      DO 360 J=2,IB1
      L=L+1
      IF (AL(I,J).EQ.0.) GO TO 360
      C=AL(I,J)*C1
      KL=0
      DO 350 K=J,IB1
      KL=KL+1
      IF (AL(I,K).NE.0.) AL(L,KL)=AL(L,KL)-C*AL(I,K)
  350 CONTINUE
      AL(I,J)=C
  360 CONTINUE
  370 CONTINUE
C**MODIFY RHS, UPPER HALF
  380 DO 400 I=1,ICR1
      JJ=IC(I,1)
      DO 390 J=2,JJ
      LR=IC(I,J)
      B(LR)=B(LR)-AU(I,J)*B(I)
  390 CONTINUE
  400 B(I)=B(I)/AU(I,1)
C**MODIFY RHS, LOWER HALF
      DO 420 I=1,LH
      IR=I+ICR1
      LR=IR
      DO 410 J=2,IB1
      LR=LR+1
      IF (AL(I,J).NE.0.) B(LR)=B(LR)-AL(I,J)*B(IR)
  410 CONTINUE
  420 B(IR)=B(IR)/AL(I,1)
C*****BACK SOLVE--LOWER HALF
      B(NEQ)=B(NEQ)/AL(NEQ-ICR1,1)
      DO 440 I=1,LH
      K=NEQ-I
```

```
      KL=K-ICR1                                                         D4 3030
      L=K                                                               D4 3040
      DO 430 J=2,IR1                                                    D4 3050
      L=L+1                                                             D4 3060
      IF (AL(KL,J).NE.0.) B(K)=B(K)-AL(KL,J)*B(L)                       D4 3070
  430 CONTINUE                                                          D4 3080
  440 CONTINUE                                                          D4 3090
C*****BACK SOLVE--UPPER HALF                                            D4 3100
      DO 460 I=1,ICR1                                                   D4 3110
      K=ICR-I                                                           D4 3120
      JJ=IC(K,1)                                                        D4 3130
      DO 450 J=2,JJ                                                     D4 3140
      L=IC(K,J)                                                         D4 3150
      B(K)=B(K)-AU(K,J)*B(L)                                            D4 3160
  450 CONTINUE                                                          D4 3170
  460 CONTINUE                                                          D4 3180
C*****COMPUTE NEW PHI VALUES                                            D4 3190
      DO 470 I=1,IM                                                     D4 3200
      DO 470 J=1,JM                                                     D4 3210
      IF (IN(I,J).EQ.0) GO TO 470                                       D4 3220
      N=I+1+DIML*J                                                      D4 3230
      IF (ITYPE.NE.1) PHE(N)=KEEP(N)                                    D4 3240
      L=IN(I,J)                                                         D4 3250
      TCHK=ABS(B(L))                                                    D4 3260
      IF (TCHK.GT.BIGI) BIGI=TCHK                                       D4 3270
      PHI(N)=PHI(N)+HMAX*B(L)                                           D4 3280
  470 CONTINUE                                                          D4 3290
C*****CHECK TERMINATION CONDITIONS                                      D4 3300
      TEST3(KOUNT+1)=BIGI                                               D4 3310
      IF (LENGTH.GT.0.AND.WATER.NE.CHK(2)) GO TO 490                    D4 3320
      IF (WATER.NE.CHK(2)) RETURN                                       D4 3330
      IF (KOUNT.GE.LENGTH.AND.BIGI.LE.ERR) RETURN                       D4 3340
      KOUNT=KOUNT+1                                                     D4 3350
      IF (KOUNT.LE.ITMAX) GO TO 480                                     D4 3360
      WRITE (P,500)                                                     D4 3370
      CALL TRANS                                                        D4 3380
      CALL TERM1                                                        D4 3390
      RETURN                                                            D4 3400
  480 CALL TRANS                                                        D4 3410
      GO TO 100                                                         D4 3420
  490 IF (KOUNT.GE.LENGTH.AND.BIGI.LE.ERR) RETURN                       D4 3430
      KOUNT=KOUNT+1                                                     D4 3440
      IF (KOUNT.LE.ITMAX) GO TO 100                                     D4 3450
      WRITE (P,500)                                                     D4 3460
      CALL TERM1                                                        D4 3470
      RETURN                                                            D4 3480
C                                                                       D4 3490
C                                                                       D4 3500
  500 FORMAT ('0EXCEEDED PERMITTED NUMBER OF ITERATIONS FOR NON-LINEAR SD4 3510
     1OLUTION'/' ',63('*'))                                             D4 3520
  510 FORMAT (1H-,41X,'SOLUTION BY LDU FACTORIZATION ASSUMING D4 ORDERIND4 3530
     1G',/,42X,50(1H_),//,61X,'BETA =',F5.2,//,45X,'ITERATIONS:  MINIMUMD4 3540
     2 =',I5,/,58X,'MAXIMUM =',I5,/,60X,'THETA =',F5.2)                 D4 3550
  520 FORMAT (1H-,25X,'*****WARNING*****MINIMUM DIMENSIONS FOR ARRAYS USD4 3560
     1ED BY THIS METHOD ARE AS FOLLOWS:',//,64X,'AU:',I5,' BY    5',/,64D4 3570
     2X,'AL:',I5,' BY',I5,/,64X,'IC:',I5,' BY    5',/,65X,'B:',I5,      D4 3580
     3/,64X,'IN:',I5,' BY',I5)                                          D4 3585
  530 FORMAT (8F10.4)                                                   D4 3590
      END                                                               D4 3600-
//
```