Techniques of Water-Resources Investigations of the United States Geological Survey

Chapter A1

A MODULAR THREE-DIMENSIONAL FINITE-DIFFERENCE GROUND-WATER FLOW MODEL

By Michael G. McDonald and Arlen W. Harbaugh

This chapter supersedes U.S. Geological Survey Open-File Report 83–875

Book 6

MODELING TECHNIQUES

CHAPTER 13

SLICE-SUCCESSIVE OVERRELAXATION PACKAGE

Conceptualization and Implementation

Successive overrelaxation is another method for solving large systems of linear equations by means of iteration. It is implemented in the model discussed herein through the Slice Successive Overrelaxation (SSOR) Package. Background material on the successive overrelaxation approach can be found in many standard references, including those already noted by Peaceman (1977), Crichlow (1977) and Remson, Hornberger and Molz (1971).

The successive overrelaxation technique is implemented in the SSOR Package by dividing the finite difference grid into vertical "slices," as shown in figure 54, and grouping the node equations into discrete sets, each set corresponding to a slice. In every iteration, these sets of equations are processed in turn, resulting in a new set of estimated head values for each slice. As the equations for each slice are processed, they are first expressed in terms of the change in computed head between successive iterations. The set of equations corresponding to the slice is then solved directly by Gaussian elimination, treating the terms for adjacent slices as known quantities (that is, inserting the most recently computed values of head for the adjacent slices as "known" values in the equations for the slice being processed). The values of head change computed for the slice in this Gaussian elimination process are then each multiplied by an acceleration parameter, ω, generally taken between 1 and 2; the results are taken as the final values of head change in that iteration

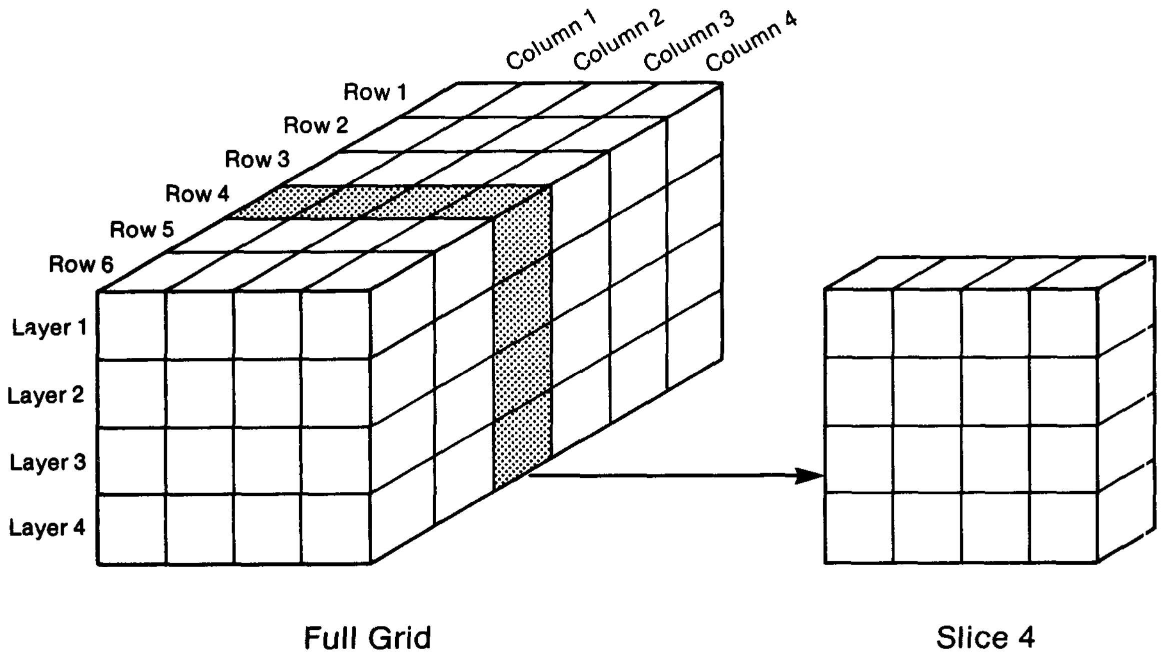


Figure 54.—Division of the three-dimensional model array into vertical slices for processing in the SSOR package.

for the slice. They are added to the respective head values from the preceding iteration to obtain the final estimates of head for the iteration, for that slice. This procedure is repeated for each slice in sequence until all of the slices in the three-dimensional array have been processed, thus completing a single iteration. The entire sequence is then repeated, in successive passes through the series of slices, until the differences between the head values computed in successive iterations is less than the closure criterion at all nodes in the mesh.

It should be noted that even though a direct method of solution (Gaussian elimination) is used within each iteration to process the equations for each individual slice, the overall solution procedure is not direct but iterative. Each direct solution produces only interim values or estimates of head change based on the most recently computed heads in adjacent slices; as successive slices are processed, the computed values continue to change until closure is achieved.

The process of solution described above can be illustrated in more detail through consideration of the node equations. The equation of flow for an individual cell, as developed in chapter 2, is reproduced below with the addition of a second superscript to indicate iteration level

$$\begin{aligned}
&CV_{i,j,k-1/2}h^{m,\ell}_{i,j,k-1} + CC_{i-1/2,j,k}h^{m,\ell}_{i-1,j,k} + CR_{i,j-1/2,k}h^{m,\ell}_{i,j-1,k} \\
&+ (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} - CR_{i,j+1/2,k} \\
&- CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})h^{m,\ell}_{i,j,k} + CR_{i,j+1/2,k}h^{m,\ell}_{i,j+1,k} \\
&+ CC_{i+1/2,j,k}h^{m,\ell}_{i+1,j,k} + CV_{i,j,k+1/2}h^{m,\ell}_{i,j,k+1} = RHS_{i,j,k} \qquad (113)
\end{aligned}$$

In equation (113), the superscript m refers to the time step, while the superscript ℓ refers to the iteration level. If an equation of the form of (113) is written for the following iteration level, ℓ+1, and the left side of equation (113) is then subtracted from each side of the new equation, the result can be written as

$$\begin{aligned}
&CV_{i,j,k-1/2},(h^{m,\ell+1}_{i,j,k-1}-h^{m,\ell}_{i,j,k-1}) + CC_{i-1/2,j,k}(h^{m,\ell+1}_{i-1,j,k}-h^{m,\ell}_{i-1,j,k}) \\
&+ CR_{i,j-1/2,k}(h^{m,\ell+1}_{i,j-1,k}-h^{m,\ell}_{i,j-1,k}) + (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} \\
&- CR_{i,j-1/2,k} - CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} \\
&+ HCOF_{i,j,k})(h^{m,\ell+1}_{i,j,k}-h^{m,\ell}_{i,j,k}) + CR_{i,j+1/2,k}(h^{m,\ell+1}_{i,j+1,k}-h^{m,\ell}_{i,j+1,k}) \\
&+ CC_{i+1/2,j,k}(h^{m,\ell+1}_{i+1,j,k}-h^{m,\ell}_{i+1,j,k}) + CV_{i,j,k+1/2}(h^{m,\ell+1}_{i,j,k+1}-h^{m,\ell}_{i,j,k+1}) = \\
&RHS_{i,j,k} - CV_{i,j,k-1/2}h^{m,\ell}_{i,j,k-1} - CC_{i-1/2,j,k}h^{m,\ell}_{i-1,j,k} \\
&- CR_{i,j-1/2,k}h^{m,\ell}_{i,j-1,k} - (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} \\
&- CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})h^{m,\ell}_{i,j,k} \\
&- CR_{i,j+1/2,k}h^{m,\ell}_{i,j+1,k} - CC_{i+1/2,j,k}h^{m,\ell}_{i+1,j,k} - CV_{i,j,k+1/2}h^{m,\ell}_{i,j,k+1}
\end{aligned} \qquad (114)$$

In equation (114) the unknown terms are taken as the changes in computed head between iteration ℓ and iteration ℓ+1--for example, $(h^{m,\ell+1}_{i,j,k} - h^{m,\ell}_{i,j,k})$. Note that when the ℓth iteration has been completed, the right hand side of (114) consists entirely of known terms--it includes the RHS and conductance terms assembled in the formulation process, and estimates of head already obtained during iteration ℓ.

Now suppose that we divide the mesh into vertical slices taken along rows, as shown in figure 54, and isolate the equations associated with the nodes of an individual slice--for example, slice 4 of figure 54, which is taken along row 4 of the three dimensional array. In terms of equation (114), if we are processing slice i, corresponding to row i, we retain the head changes at nodes within this slice as unknown terms, but consider the head changes at nodes in the two adjacent slices to be known values. Thus the terms $CC_{i-1/2,j,k}\,(h^{m,\ell+1}_{i-1,j,k} - h^{m,\ell}_{i-1,j,k})$ and $CC_{i+1/2,j,k}\,(h^{m,\ell+1}_{i+1,j,k} - h^{m,\ell}_{i+1,j,k})$, on the left side of equation (114), are treated as known quantities. If we move these two expressions to the right side of the equation and rearrange, we find that the terms in $h^{m,\ell}_{i-1,j,k}$ and and $h^{m,\ell}_{i+1,j,k}$ drop out, leaving

$$CV_{i,j,k-1/2}\,(h^{m,\ell+1}_{i,j,k-1}-h^{m,\ell}_{i,j,k-1}) + CR_{i,j-1/2,k}(h^{m,\ell+1}_{i,j-1,k}-h^{m,\ell}_{i,j-1,k})$$

$$+ (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} - CR_{i,j+1/2,k}$$

$$- CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})(h^{m,\ell+1}_{i,j,k}-h^{m,\ell}_{i,j,k})$$

$$+ CR_{i,j+1/2,k}(h^{m,\ell+1}_{i,j+1,k}-h^{m,\ell}_{i,j+1,k}) + CV_{i,j,k+1/2}(h^{m,\ell+1}_{i,j,k+1}-h^{m,\ell}_{i,j,k+1}) =$$

$$RHS_{i,j,k} - CV_{i,j,k-1/2}h^{m,\ell}_{i,j,k-1} - CC_{i-1/2,j,k}h^{m,\ell+1}_{i-1,j,k}$$

$$- CR_{i,j-1/2,k}h^{m,\ell}_{i,j-1,k} - (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k}$$

$$- CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})h^{m,\ell}_{i,j,k}$$

$$- CR_{i,j+1/2,k}h^{m,\ell}_{i,j+1,k} - CC_{i+1/2,j,k}h^{m,\ell+1}_{i+1,j,k} - CV_{i,j,k+1/2}h^{m,\ell}_{i,j,k+1}$$

(115)

Now suppose the slices are processed in the order of increasing row number, i; then calculations for slice i-1 will be completed in each iteration before calculations for slice i are initiated. It follows that a value of $h^{m,\ell+1}_{i-1,j,k}$ will be available when the processing of slice i is initiated in iteration $\ell+1$, whereas a value of $h^{m,\ell+1}_{i+1,j,k}$ will not be available. Thus the term $CC_{i-1/2,j,k}h^{m,\ell+1}_{i-1,j,k}$ can be incorporated directly as a known term in the processing of slice i, but the term $CC_{i+1/2,j,k}h^{m,\ell+1}_{i+1,j,k}$ cannot. To circumvent this difficulty, the value of $h^{m}_{i+1,j,k}$ from the preceding iteration, $h^{m,\ell}_{i+1,j,k}$, is substituted for $h^{m,\ell+1}_{i+1,j,k}$ on the right side of (115). (Thus in effect we are using the most recently calculated value of head for each adjacent slice.) The resulting equation is

$$\begin{aligned}
&CV_{i,j,k-1/2}(\tilde{h}^{m,\ell+1}_{i,j,k-1}-h^{m,\ell}_{i,j,k-1}) + CR_{i,j-1/2,k}(\tilde{h}^{m,\ell+1}_{i,j-1,k}-h^{m,\ell}_{i,j-1,k}) \\
&+ (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} - CR_{i,j+1/2,k} \\
&- CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})(\tilde{h}^{m,\ell+1}_{i,j,k}-h^{m,\ell}_{i,j,k}) \\
&+ CR_{i,j+1/2,k}(\tilde{h}^{m,\ell+1}_{i,j+1,k}-h^{m,\ell}_{i,j+1,k}) + CV_{i,j,k+1/2}(\tilde{h}^{m,\ell+1}_{i,j,k+1}-h^{m,\ell}_{i,j,k+1}) = \\
&RHS_{i,j,k} - CV_{i,j,k-1/2}h^{m,\ell}_{i,j,k-1} - CC_{i-1/2,j,k}h^{m,\ell+1}_{i-1,j,k} \\
&- CR_{i,j-1/2,k}h^{m,\ell}_{i,j-1,k} - (- CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} \\
&- CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})h^{m,\ell}_{i,j,k} \\
&- CR_{i,j+1/2,k}h^{m,\ell}_{i,j+1,k} - CC_{i+1/2,j,k}h^{m,\ell}_{i+1,j,k} - CV_{i,j,k+1/2}h^{m,\ell}_{i,j,k+1}
\end{aligned} \qquad (116)$$

In equation (116), the notation $\tilde{h}$ has been introduced for the head terms in slice i at iteration $\ell+1$. The purpose of this notation will become clear as the solution process is described. The number of nodes in the slice is NC*NL, where NC is the number of columns in the model and NL the number of layers; and an equation of the form of (116) is formulated at each node. Thus a system of NC*NL equations in NC*NL unknowns is established. Because the number of layers is usually small, the total number of equations is generally small enough so that direct solution by Gaussian elimination is an efficient approach (note that such a procedure would generally not be feasible for the larger set of equations associated with the entire three-dimensional model array.)

The set of equations associated with an individual slice, i, can be written in matrix form as

$$[A]_i\{\Delta\tilde{h}\}_i = \{R\}_i \tag{117}$$

where $[A]_i$ is the coefficient matrix for slice i; $\{\Delta\tilde{h}\}_i$ is a vector of estimates, $\tilde{h}^{m,\ell+1}_{i,j,k}-h^{m,\ell}_{i,j,k}$, for the change in computed head at each node in the slice between iteration ℓ and iteration $\ell+1$; and $\{R\}_i$ is the vector of "constant" terms, representing the right side of equation (116), for slice i.

The Gaussian elimination procedure applied to the matrix equations (117) yields one value of the term $(\tilde{h}^{m,\ell+1}_{i,j,k}-h^{m,\ell}_{i,j,k})$ for each node in the slice. These terms are taken as first estimates for the change in computed head from iteration ℓ to iteration ℓ +1. Each is multiplied by the acceleration parameter, ω, and each result is added to the corresponding head from the preceding iteration to obtain the final estimate of head for iteration ℓ+1; that is,

$$h^{m,\ell+1}_{i,j,k} = h^{m,\ell}_{i,j,k} + \omega(\tilde{h}^{m,\ell+1}_{i,j,k}-h^{m,\ell}_{i,j,k}) \tag{118}$$

When values of $h_{i,j,k}^{m,\ell+1}$ have been computed for each node (j,k) in slice i, the procedure of calculation is initiated for the succeeding slice, i+1. When all slices have been processed the iteration is complete, and calculations are initiated for the next iteration unless closure has been achieved.

As illustrated in figure 55-a, the matrix of coefficients $[A]_i$ of equation (117) is symmetric and banded, with a maximum half-bandwidth equal to the number of layers. Because of the symmetry of the matrix, only the lower triangular portion has to be stored; this storage is provided in the program in a two-dimensional array, as illustrated in figure 55-b, with dimensions NL*NC and NL+1. In this example, NL=NC=3.

Adjustment of the acceleration parameter is frequently necessary in SSOR to achieve optimal rates of convergence. For this purpose, methods similar to the trial and error procedure described in Chapter 12, for adjustment of the SIP "seed" value can be applied.

a_{11}	a_{12}		a_{14}					
a_{12}	a_{22}	a_{23}		a_{25}				
	a_{23}	a_{33}			a_{36}			
a_{14}			a_{44}	a_{45}		a_{47}		
	a_{25}		a_{45}	a_{55}	a_{56}			
		a_{36}		a_{56}	a_{66}		a_{68}	
			a_{47}			a_{77}		
					a_{68}		a_{88}	a_{89}
							a_{89}	a_{99}

(a) Coefficient matrix for an individual slice

a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}	a_{88}	a_{99}
a_{12}	a_{23}		a_{45}	a_{56}			a_{89}	
					a_{68}			
a_{14}	a_{25}	a_{36}	a_{47}					

(b) Two dimensional array for storage of matrix elements

Figure 55.—Coefficient matrix for slice equations and corresponding computer storage array.

Slice-Successive Overrelaxation Package Input

Input to the Slice-Successive Overrelaxation (SOR) Package is read from the unit specified in IUNIT(11).

FOR EACH SIMULATION

SOR1AL

```
1. Data:   MXITER
   Format: I10
```

SOR1RP

```
2. Data:   ACCL    HCLOSE  IPRSOR
   Format: F10.0   F10.0   I10
```

Explanation of Fields Used in Input Instructions

MXITER--is the maximum number of iterations allowed in a time step.

ACCL--is the acceleration parameter, usually between 1.0 and 2.0.

HCLOSE--is the head change criterion for convergence. When the maximum absolute value of head change from all nodes during an iteration is less than or equal to HCLOSE, iteration stops.

IPRSOR--is the printout interval for SOR. IF IPRSOR is equal to zero, it is changed to 999. The maximum head change (positive or negative) is printed for each iteration of a time step whenever the time step is an even multiple of IPRSOR. This printout also occurs at the end of each stress period regardless of the value of IPRSOR.

Module Documentation for the Slice-Successive Overrelaxation Package

The Slice-Successive Overrelaxation Package (SOR1) consists of three primary modules and one submodule. They are:

Primary Modules

SOR1AL	Allocates space for arrays.
SOR1RP	Reads control information needed by the SOR1 Package.
SOR1AP	Performs one iteration of slice-successive overrelaxation.

Submodule

SSOR1B	Solves a system of linear equations.

Narrative for Module SOR1AL

Module SOR1AL allocates space in the X array for SOR arrays. The SOR arrays are A, RES, IEQPNT, HDCG, and LRCH. "A" holds the main diagonal and the lower diagonals of the symmetric coefficient matrix for a single slice. RES holds the residual vector (the right hand sides) for a single slice. IEQPNT holds a sequential identification number for each cell in a slice. HDCG holds the maximum head change for each iteration. LRCH holds the location of the cell (row, column, and layer) which had the maximum head change for each iteration.

Module SOR1AL performs its functions in the following order:

1. Print a message identifying the SOR Package.

2. Read and print the maximum number of iterations.

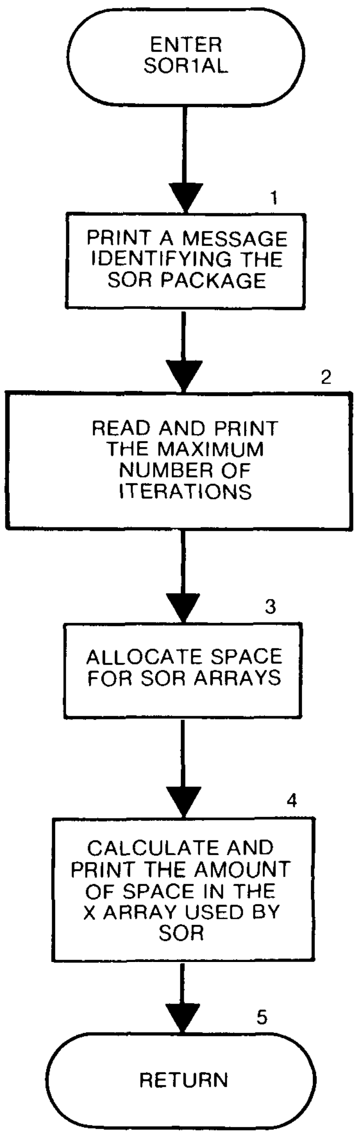
3. Allocate the required space in the X array. The X-array location pointer (ISUM) is saved in the variable ISOLD prior to allocation so that the space required for SOR can be calculated in step 4. To allocate space for an array, the array-location variable is set equal to ISUM. Then ISUM is incremented by the required number of elements.

4. Calculate and print the space used in the X array. The space used by SOR is ISUM - ISOLD.

5. RETURN

Flow Chart for Module SOR1AL

X array is the pool of memory space from which space is allocated for arrays used by various packages.


ENTER SOR1AL
1
PRINT A MESSAGE IDENTIFYING THE SOR PACKAGE
2
READ AND PRINT THE MAXIMUM NUMBER OF ITERATIONS
3
ALLOCATE SPACE FOR SOR ARRAYS
4
CALCULATE AND PRINT THE AMOUNT OF SPACE IN THE X ARRAY USED BY SOR
5
RETURN

```
      SUBROUTINE SOR1AL(ISUM,LENX,LCA,LCRES,LCHDCG,LCLRCH,LCIEQP,
     1       MXITER,NCOL,NLAY,NSLICE,MBW,IN,IOUT)
C
C-----VERSION 1638 24JUL1987 SOR1AL
C     ******************************************************************
C     ALLOCATE STORAGE FOR SOR ARRAYS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C     ------------------------------------------------------------------
C
C1------PRINT A MESSAGE IDENTIFYING SOR PACKAGE
      WRITE(IOUT,1)IN
    1 FORMAT(1H0,'SOR1 -- SLICE-SUCCESSIVE OVERRELAXATION PACKAGE'
     1,', VERSION 1, 9/1/87 INPUT READ FROM UNIT',I3)
C
C2------READ AND PRINT MXITER (MAXIMUM # OF ITERATIONS)
      READ(IN,2) MXITER
    2 FORMAT(I10)
      WRITE(IOUT,3) MXITER
    3 FORMAT(1X,I5,' ITERATIONS ALLOWED FOR SOR CLOSURE')
C
C3------ALLOCATE SPACE FOR THE SOR ARRAYS
      ISOLD=ISUM
      NSLICE=NCOL*NLAY
      MBW=NLAY+1
      LCA=ISUM
      ISUM=ISUM+NSLICE*MBW
      LCRES=ISUM
      ISUM=ISUM+NSLICE
      LCIEQP=ISUM
      ISUM=ISUM+NSLICE
      LCHDCG=ISUM
      ISUM=ISUM+MXITER
      LCLRCH=ISUM
      ISUM=ISUM+3*MXITER
      ISP=ISUM-ISOLD
C
C4------CALCULATE AND PRINT THE SPACE USED IN THE X ARRAY
      WRITE(IOUT,4) ISP
    4 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE USED BY SOR')
      ISUM1=ISUM-1
      WRITE(IOUT,5) ISUM1,LENX
    5 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I8)
      IF(ISUM1.GT.LENX) WRITE(IOUT,6)
    6 FORMAT(1X,'   ***X ARRAY MUST BE DIMENSIONED LARGER***')
C
C5------RETURN
      RETURN
      END
```

List of Variables for Module SOR1AL

Variable	Range	Definition
IN	Package	Primary unit number from which input for this package will be read.
IOUT	Global	Primary unit number for all printed output. IOUT = 6.
ISOLD	Package	Before this module allocates space, ISOLD is set equal to ISUM. After allocation, ISOLD is subtracted from ISUM to get ISP, the amount of space in the X array allocated by this module.
ISP	Module	Number of words in the X array allocated by this module.
ISUM	Global	Index number of the lowest element in the X array which has not yet been allocated. When space is allocated for an array, the size of the array is added to ISUM.
ISUM1	Module	Index number of the last element of the X array allocated by this module.
LCA	Package	Location in the X array of the first element of array A.
LCHDCG	Package	Location in the X array of the first element of array HDCG.
LCIEQP	Package	Location in the X array of the first element of array IEQPNT.
LCLRCH	Package	Location in the X array of the first element of array LRCH.
LCRES	Package	Location in the X array of the first element of array RES.
LENX	Global	Length of the X array in words. This should always be equal to the dimension of X specified in the MAIN program.
MBW	Package	Maximum bandwidth of the coefficient matrix +1.
MXITER	Package	Maximum number of iterations.
NCOL	Global	Number of columns in the grid.
NLAY	Global	Number of layers in the grid.
NSLICE	Package	Number of cells in a slice.

Narrative for Module SOR1RP

Module SOR1RP reads data for the SOR package: the acceleration parameter (ACCL), also called the relaxation factor; the closure criterion (HCLOSE); and the time-step interval (IPRSOR) for printing head change. This module does not have a flow chart. Module SOR1RP performs its functions in the following order:

1. Read the acceleration parameter (ACCL), the closure criterion (HCLOSE), and the interval for printing head change (IPRSOR). If ACCL is zero, substitute a default value of 1.0. If IPRSOR is less than one, set it equal to 999.

2. Print the maximum number of iterations (MXITER), the acceleration parameter (ACCL), the closure criterion (HCLOSE), and the head-change interval (IPRSOR).

3. RETURN.

```
      SUBROUTINE SOR1RP(MXITER,ACCL,HCLOSE,IN,IPRSOR,IOUT)
C
C
C-----VERSION 1005 16MAR1983 SOR1RP
C     ******************************************************************
C     READ PARAMETERS FOR SOR
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C     ------------------------------------------------------------------
C
C1------READ THE ACCELERATION PARAMETER/RELAXATION FACTOR (ACCL) THE
C1------CLOSURE CRITERION (HCLOSE) AND THE NUMBER OF TIME STEPS
C1------BETWEEN PRINTOUTS OF MAXIMUM HEAD CHANGES (IPRSOR).
      READ(IN,1) ACCL,HCLOSE,IPRSOR
    1 FORMAT(2F10.0,I10)
      IF(ACCL.EQ.0.) ACCL=1.
      IF(IPRSOR.LT.1) IPRSOR=999
C
C2------PRINT ACCL, HCLOSE, IPRSOR
      WRITE(IOUT,100)
  100 FORMAT(1H0,///57X,'SOLUTION BY SLICE-SUCCESSIVE OVERRELAXATION'
     1/57X,43('-'))
      WRITE(IOUT,115) MXITER
  115 FORMAT(1H0,47X,'MAXIMUM ITERATIONS ALLOWED FOR CLOSURE =',I9)
      WRITE(IOUT,120) ACCL
  120 FORMAT(1H ,63X,'ACCELERATION PARAMETER =',G15.5)
      WRITE(IOUT,125) HCLOSE
  125 FORMAT(1H ,52X,'HEAD CHANGE CRITERION FOR CLOSURE =',E15.5)
      WRITE(IOUT,130) IPRSOR
  130 FORMAT(1H ,52X,'SOR HEAD CHANGE PRINTOUT INTERVAL =',I9)
C
C3------RETURN
      RETURN
      END
```

List of Variables for Module SOR1RP

Variable	Range	Definition
ACCL	Package	Acceleration parameter.
HCLOSE	Package	Closure criterion for the iterative procedure.
IN	Package	Primary unit number from which input for this package will be read.
IOUT	Global	Primary unit number for all printed output. IOUT = 6.
IPRSOR	Package	Frequency (in time steps) with which the maximum head changes for each iteration will be printed.
MXITER	Package	Maximum number of iterations.

Narrative for Module SOR1AP

Module SOR1AP performs one iteration of the Slice-Successive Over-relaxation (SSOR) algorithm for solving the system of finite-difference equations. The conductances CC, CR, and CV and the composite terms HCOF and RHS (see equation (27)) which are calculated by the formulation procedure are combined, row by row (slice by slice), to form the coefficient matrix [A] and the vector $\{RES\}$ on the right hand side of the matrix equation for a single slice. Since the coefficient matrix is symmetric and banded, only main diagonals and NLAY subdiagonals are saved. As heads are calculated, they are stored in the array HNEW. The matrix [A] and the vector $\{RES\}$ are passed to a submodule SSOR1B which solves the matrix equation for a vector of approximate head changes which is then multiplied by the relaxation factor to get the final head changes for the iteration. The final head changes are added to the heads from the preceding iteration to get the heads for the current iteration. The final head changes for the iteration are compared to the closure criterion to see if the iterative procedure has closed.

Module SOR1AP performs its functions in the following order:

1. Calculate the number of elements in the compressed coefficient matrix [A].

2. Process the slices (rows) one at a time (DO STEPS 3-7).

3. Clear the A array.

4. Assign integers sequentially to the active cells in the slice (remember that finite-difference equations are formulated only for active cells).

5. Calculate the elements in the compressed coefficient matrix [A] and the residual vector {RES}. Process the cells in the slice one cell at a time.

If the cell is inactive, move on to the next cell. The elements in the main diagonal of the coefficient matrix (the multipliers of $h_{i,j,k}$) will consist of HCOF plus conductances to the six adjacent cells. They will be formed in an accumulator called EE. The contents of EE multiplied by the head from the previous iteration (HNEW) are subtracted from an accumulator (R) to form the residual.

(a) Determine the equation number (NEQ) of the cell. If NEQ is zero, the cell is inactive. Move on to the next cell.

(b) Set the accumulators EE and R equal to HCOF and RHS, respectively. Note: HNEW contains head from the last iteration.

(c) If there is a node to the left, subtract the conductance from EE and subtract the conductance times HNEW from R.

(d) If there is a node to the right, subtract the conductance from EE, and subtract the conductance times HNEW from R; and, if the cell to the right is active, move the conductance into the compressed coefficient matrix [A]. Remember that the coefficient matrix is symmetric so the conductance to the left in step 5(c) did not have to be stored.

(e) If there is a node to the rear, subtract the conductance from EE and subtract the conductance times HNEW from R.

(f) If there is a node to the front, subtract the conductance from EE and subtract the conductance times HNEW from R. Remember that the form of the SSOR equations does not have terms containing head in adjacent rows on the left hand side.

(g) If there is a node above, subtract the conductance from EE and subtract the conductance times HNEW from R.

(h) If there is a node below, subtract the conductance from EE and subtract the conductance times HNEW from R; and, if the cell below is active, move the conductance into A.

(i) Move EE into the first row of A. The first row in A corresponds to the main diagonal in the "full" coefficient matrix. Subtract EE times HNEW from R and store it in the residual vector.

6. If there are no equations for this slice, go on to the next slice. If there is only one equation, solve it directly and leave the result in the residual vector {RES}. If there are two or more equations, call submodule SSOR1B to solve the system of equations for the slice leaving the results (first estimate of head change for this iteration) in the vector {RES}.

7. For each cell in the slice, calculate the head for the current iteration.

(a) Multiply the first estimate of head change for this iteration by the relaxation factor to get the final estimate of head change for this iteration.

(b) Add the final head change for this iteration to the head from the last iteration to get the head for this iteration.

(c) If the head change for this cell is greater than that for any other cell, store the head change and the location of the cell.

8. Save the largest head change from this iteration so that it can be printed at the end of the time step.

9. Compare the biggest head change (BIGG) to the closure criterion (HCLOSE). If HCLOSE is greater than BIGG, set the convergence flag (ICNVG) equal to one.

10. If you have not converged and you have not exceeded the maximum number of iterations, RETURN.

11. Print the number of iterations.

12. If convergence failed, or this is the last time step, or this is the time step interval specified by the user, print the maximum head change for each iteration in this time step.

13. RETURN.

Flow Chart for Module SOR1AP

A is a compressed coefficient matrix for a slice. It contains the main diagonal of the full matrix and the NLAY diagonals below it. (NLAY is the number of layers.)

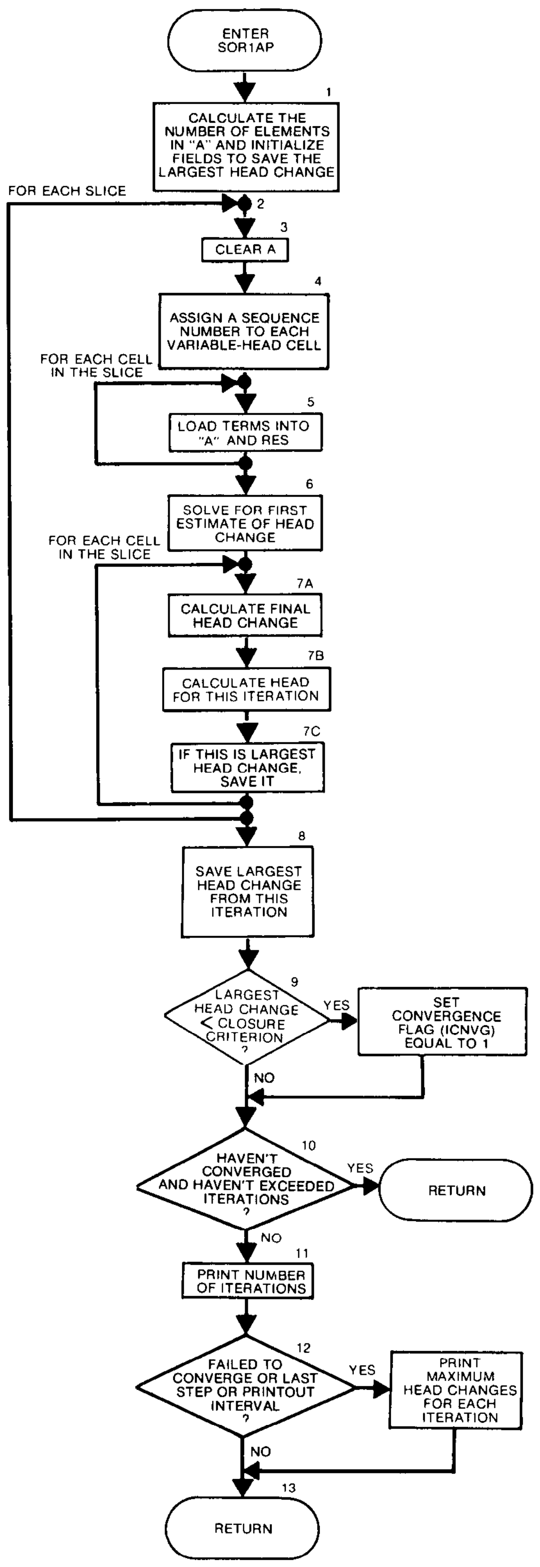
Sequence Number is a number used to identify the internal (variable-head) cells in a slice and also the equations for each internal cell.

RES is a vector containing the residuals for a slice. It consists of RHS (from the basic finite-difference equation) plus all of those terms which are moved to the right hand side to get the equations ready for solution in residual form.

First estimates of head change: these are the head changes calculated by simultaneously solving the equations for a slice. They will be multiplied by the relaxation factor to get final estimates of head change.

Final estimates of head change: these are the head changes calculated by multiplying first estimates by the relaxation factor. They are added to the heads from the previous iteration to get head for the current iteration.

ICNVG is the convergence flag. It is set in the approximator and returned to the MAIN Program so that the iteration loop can be terminated.


ENTER SOR1AP
1
CALCULATE THE NUMBER OF ELEMENTS IN "A" AND INITIALIZE FIELDS TO SAVE THE LARGEST HEAD CHANGE
FOR EACH SLICE
2
3
CLEAR A
4
ASSIGN A SEQUENCE NUMBER TO EACH VARIABLE-HEAD CELL
FOR EACH CELL IN THE SLICE
5
LOAD TERMS INTO "A" AND RES
6
SOLVE FOR FIRST ESTIMATE OF HEAD CHANGE
FOR EACH CELL IN THE SLICE
7A
CALCULATE FINAL HEAD CHANGE
7B
CALCULATE HEAD FOR THIS ITERATION
7C
IF THIS IS LARGEST HEAD CHANGE, SAVE IT
8
SAVE LARGEST HEAD CHANGE FROM THIS ITERATION
9
LARGEST HEAD CHANGE < CLOSURE CRITERION ?
YES
SET CONVERGENCE FLAG (ICNVG) EQUAL TO 1
NO
10
HAVEN'T CONVERGED AND HAVEN'T EXCEEDED ITERATIONS ?
YES
RETURN
NO
11
PRINT NUMBER OF ITERATIONS
12
FAILED TO CONVERGE OR LAST STEP OR PRINTOUT INTERVAL ?
YES
PRINT MAXIMUM HEAD CHANGES FOR EACH ITERATION
NO
13
RETURN

```
      SUBROUTINE SOR1AP(HNEW,IBOUND,CR,CC,CV,HCOF,RHS,A,RES,IEQPNT,
     1       HDCG,LRCH,KITER,HCLOSE,ACCL,ICNVG,KSTP,KPER,
     2       IPRSOR,MXITER,NSTP,NCOL,NROW,NLAY,NSLICE,MBW,IOUT)
C-----VERSION 0936 09MAY1983 SOR1AP
C     ******************************************************************
C     SOLUTION BY SLICE-SUCCESSIVE OVERRELAXATION -- 1 ITERATION
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      DOUBLE PRECISION HNEW,DIFF,DP,EE,R,HCFHNW,HHCOF
C
      DIMENSION HNEW(NCOL,NROW,NLAY), IBOUND(NCOL,NROW,NLAY),
     1   CR(NCOL,NROW,NLAY), CC(NCOL,NROW,NLAY),
     1   CV(NCOL,NROW,NLAY), HCOF(NCOL,NROW,NLAY), RHS(NCOL,NROW,NLAY),
     2   HDCG(MXITER), LRCH(3,MXITER),A(MBW,NSLICE),RES(NSLICE),
     3   IEQPNT(NLAY,NCOL)
C     ------------------------------------------------------------------
C
C1------CALCULATE # OF ELEMENTS IN COMPRESSED MATRIX A AND
C1------INITIALIZE FIELDS TO SAVE LARGEST HEAD CHANGE.
      NA=MBW*NSLICE
      BIG=0.
      ABSBIG=0.
      IB=0
      JB=0
      KB=0
C
C2------PROCESS EACH SLICE
      DO 500 I=1,NROW
C
C3------CLEAR A
      DO 110 J=1,NSLICE
      DO 110 K=1,MBW
  110 A(K,J)=0.
C
C4------ASSIGN A SEQUENCE # TO EACH VARIABLE HEAD CELL.
      NEQT=0
      DO 200 J=1,NCOL
      DO 200 K=1,NLAY
      IEQPNT(K,J)=0
      IF(IBOUND(J,I,K).LE.0) GO TO 200
      NEQT=NEQT+1
      IEQPNT(K,J)=NEQT
  200 CONTINUE
C
C5------FOR EACH CELL LOAD MATRIX A AND VECTOR RES
      DO 300 J=1,NCOL
      DO 300 K=1,NLAY
C
C5A-----IF SEQUENCE # IS 0 (CELL IS EXTERNAL) GO ON TO NEXT CELL
      NEQ=IEQPNT(K,J)
      IF(NEQ.EQ.0) GO TO 300
C
C5B-----INITIALIZE ACCUMULATORS EE AND R
      EE=0.
      R=RHS(J,I,K)
C
```

```
C5C-----IF NODE TO LEFT SUBTRACT TERMS FROM EE AND R
      IF(J.EQ.1) GO TO 120
      DP=CR(J-1,I,K)
      R=R-DP*HNEW(J-1,I,K)
      EE=EE-DP
C
C5D-----IF NODE TO RIGHT SUBTRACT TERMS FROM EE & R, MOVE COND TO A
  120 IF(J.EQ.NCOL) GO TO 125
      SP=CR(J,I,K)
      DP=SP
      R=R-DP*HNEW(J+1,I,K)
      EE=EE-DP
      NXT=IEQPNT(K,J+1)
      IF(NXT.GT.0) A(1+NXT-NEQ,NEQ)=SP
C
C5E-----IF NODE TO REAR SUBTRACT TERMS FROM EE AND R
  125 IF(I.EQ.1) GO TO 130
      DP=CC(J,I-1,K)
      R=R-DP*HNEW(J,I-1,K)
      EE=EE-DP
C
C5F-----IF NODE TO FRONT SUBTRACT TERMS FROM EE AND R
  130 IF(I.EQ.NROW) GO TO 132
      DP=CC(J,I,K)
      R=R-DP*HNEW(J,I+1,K)
      EE=EE-DP
C
C5G-----IF NODE ABOVE SUBTRACT TERMS FROM EE AND R
  132 IF(K.EQ.1) GO TO 134
      DP=CV(J,I,K-1)
      R=R-DP*HNEW(J,I,K-1)
      EE=EE-DP
C
C5H-----IF NODE BELOW SUBTRACT TERMS FROM EE & R AND MOVE COND TO A
  134 IF(K.EQ.NLAY) GO TO 136
      SP=CV(J,I,K)
      DP=SP
      R=R-DP*HNEW(J,I,K+1)
      EE=EE-DP
      IF(IEQPNT(K+1,J).GT.0) A(2,NEQ)=SP
C
C5I-----MOVE EE INTO A, SUBTRACT EE TIMES LAST HEAD FROM R TO GET RES
  136 HHCOF=HCOF(J,I,K)
      A(1,NEQ)=EE+HHCOF
      HNW=HNEW(J,I,K)
      HCFHNW=HNW*HCOF(J,I,K)
      RES(NEQ)=R-EE*HNEW(J,I,K)-HCFHNW
  300 CONTINUE
C
C6------IF NO EQUATIONS GO TO NEXT SLICE, IF ONE EQUATION SOLVE
C6------DIRECTLY, IF 2 EQUATIONS CALL SSOR1B TO SOLVE FOR FIRST
C6------ESTIMATE OF HEAD CHANGE FOR THIS ITERATION.
      IF(NEQT.LT.1) GO TO 500
      IF(NEQT.EQ.1) RES(1)=RES(1)/A(1,1)
      IF(NEQT.GE.2) CALL SSOR1B(A,RES,NEQT,NA,MBW)
C
C7------FOR EACH CELL IN SLICE CALCULATE FINAL HEAD CHANGE THEN HEAD.
      DO 400 J=1,NCOL
      DO 400 K=1,NLAY
      NEQ=IEQPNT(K,J)
      IF(NEQ.EQ.0) GO TO 400
C
```

```
C7A-----MULTIPLY FIRST ESTIMATE OF HEAD CHANGE BY RELAX FACTOR TO
C7A-----GET FINAL ESTIMATE OF HEAD CHANGE FOR THIS ITERATION.
      DH=RES(NEQ)*ACCL
      DIFF=DH
C
C7B-----ADD FINAL ESTIMATE TO HEAD FROM LAST ITERATION TO GET HEAD
C7B-----FOR THIS ITERATION.
      HNEW(J,I,K)=HNEW(J,I,K)+DIFF
C
C7C-----SAVE FINAL HEAD CHANGE IF IT IS THE LARGEST
      ABSDH=ABS(DH)
      IF(ABSDH.LE.ABSBIG) GO TO 400
      ABSBIG=ABSDH
      BIG=DH
      IB=I
      JB=J
      KB=K
  400 CONTINUE
C
C
  500 CONTINUE
C
C8------SAVE LARGEST HEAD CHANGE FOR THIS ITERATION
      HDCG(KITER)=BIG
      LRCH(1,KITER)=KB
      LRCH(2,KITER)=IB
      LRCH(3,KITER)=JB
C
C9------IF LARGEST HEAD CHANGE IS SMALLER THAN CLOSURE THEN SET
C9------CONVERGE FLAG (ICNVG) EQUAL TO 1.
      ICNVG=0
      IF(ABSBIG.LE.HCLOSE) ICNVG=1
C
C10-----IF NOT CONVERGED AND NOT EXCEDED ITERATIONS THEN RETURN
      IF(ICNVG.EQ.0 .AND. KITER.NE.MXITER) RETURN
      IF(KSTP.EQ.1) WRITE(IOUT,600)
  600 FORMAT(1H0)
C
C11-----PRINT NUMBER OF ITERATIONS
      WRITE(IOUT,601) KITER,KSTP,KPER
  601 FORMAT(1X,I5,' ITERATIONS FOR TIME STEP',I4,' IN STRESS PERIOD',
     1        I3)
C
C12-----IF FAILED TO CONVERGE OR LAST TIME STEP OR PRINTOUT
C12-----INTERVAL SPECIFIED BY USER IS HERE THEN PRINT MAXIMUM
C12-----HEAD CHANGES FOR EACH ITERATION.
      IF(ICNVG.NE.0 .AND. KSTP.NE.NSTP .AND. MOD(KSTP,IPRSOR).NE.0)
     1      GO TO 700
      WRITE(IOUT,5)
    5 FORMAT(1H0,'MAXIMUM HEAD CHANGE FOR EACH ITERATION:'/
     1    1H0,4('    HEAD CHANGE  LAYER,ROW,COL')/1X,120('-'))
      WRITE (IOUT,10) (HDCG(J),(LRCH(I,J),I=1,3),J=1,KITER)
   10 FORMAT((1X,4(4X,G12.4,' (',I3,',',I3,',',I3,')')))
      WRITE(IOUT,11)
   11 FORMAT(1H0)
C
C13-----RETURN
  700 RETURN
C
      END
```

List of Variables for Module SOR1AP

Variable	Range	Definition
A	Package	DIMENSION (MBW,NSLICE), Compressed coefficient matrix for a slice.
ABSDIG	Module	Largest ABSDH for this iteration.
ABSDH	Module	Absolute value of head change in a cell for the current iteration.
ACCL	Package	Acceleration parameter.
CC	Global	DIMENSION (NCOL,NROW,NLAY), Conductance in the column direction. CC(J,I,K) contains conductance between nodes (J,I,K) and (J,I+1,K).
CR	Global	DIMENSION (NCOL,NROW,NLAY), Conductance in the row direction. CR(J,I,K) contains conductance between nodes (J,I,K) and (J+1,I,K).
CV	Global	DIMENSION (NCOL,NROW,NLAY-1), Conductance in the vertical direction. CV(J,I,K) contains conductance between nodes (J,I,K) and (J,I,K+1).
DH	Module	Change in head in a cell during one iteration.
DIFF	Module	Double-precision change in head (DH).
DP	Module	Double-precision temporary field.
EE	Module	Main diagonal term in the finite-difference equation.
HCLOSE	Package	Closure criterion for the iterative procedure.
HCOF	Global	DIMENSION (NCOL,NROW,NLAY), Coefficient of head in the cell (J,I,K) in the finite-difference equation.
HDCG	Package	DIMENSION (MXITER), Maximum head change for each iteration.
HNEW	Global	DIMENSION (NCOL,NROW,NLAY), Most recent estimate of head in each cell. HNEW changes at each iteration.
I	Module	Index for rows.
IB	Module	Row number of the cell containing the largest head change.
IBOUND	Global	DIMENSION (NCOL,NROW,NLAY), Status of each cell. < 0, constant-head cell = 0, inactive cell > 0, variable-head cell
ICNVG	Global	Flag is set equal to one when the iteration procedure has converged.
IEQPNT	Global	DIMENSION (NLAY,NCOL), Sequence numbers for variable-head cells in a slice.
IOUT	Global	Primary unit number for all printed output. IOUT = 6.
IPRSOR	Package	Frequency (in time steps) with which the maximum head changes for each iteration will be printed.
J	Module	Index for columns.
JB	Module	Column number of the cell containing the largest head change.
K	Module	Index for layers.
KB	Module	Layer number of the cell containing the largest head change.
KITER	Global	Iteration counter. Reset at the start of each time step.
KPER	Global	Stress period counter.

List of Variables for Module SOR1AP (Continued)

Variable	Range	Definition
KSTP	Global	Time step counter. Reset at the start of each stress period.
LRCH	Package	DIMENSION (MXITER), Layer, row, and column of the cell containing the maximum head change (HDCG) for each iteration.
MBW	Package	Maximum bandwidth of the coefficient matrix +1.
MXITER	Package	Maximum number of iterations.
NA	Package	Number of elements in the compressed coefficient matrix (A).
NCOL	Global	Number of columns in the grid.
NEQ	Module	Index for equations (variable-head cells) in a slice.
NEQT	Package	Number of equations (variable-head cells) in a slice.
NLAY	Global	Number of layers in the grid.
NROW	Global	Number of rows in the grid.
NSLICE	Package	Number of cells in a slice.
NSTP	Global	Number of time steps in the current stress period.
NXT	Module	Sequence number of the cell to the right.
R	Module	Right hand side of the finite-difference equation as modified (terms for the adjacent rows moved to the right) for solution by the slice-successive overrelaxation.
RES	Package	DIMENSION (NSLICE), Residual.
RHS	Global	DIMENSION (NCOL,NROW,NLAY), Right hand side of the finite-difference equation. RHS is an accumulation of terms from several different packages.
SP	Module	Single-precision temporary field.