Schlumberger

Reference Manual

2008.1
# Table of Contents

**Appendix D - Sample Problems** .................................................................................................................. 131
- Key to sample problems ................................................................................................................................. 131
- Interpolation of porosity (1) ........................................................................................................................... 132
- Corner point geometry with sloping faults (2) ............................................................................................... 133
- Corner point geometry with distorted grid (3) ............................................................................................... 135
- The FILL operation applied to x coordinates (4) ......................................................................................... 136
- Conversion of block-centre depths (TOPS, DEPTH, DZ) (5) ....................................................................... 137
- Output controls to suppress ZCORN etc. (6) ............................................................................................... 139
- Radial data (7) .................................................................................................................................................. 140

**Appendix E - Index** ........................................................................................................................................ 143
List of Figures

Figure 2.1 .......... Co-ordinate lines & sloping faults ............................................................... 12
Figure 2.2 .......... Block centred & node centred geometries .............................................. 14
Figure 2.3 .......... A Wedge ................................................................................................... 18
Figure 2.4 .......... A Wedge & Gap .......................................................................................... 19
Figure 2.5 .......... A Wedge, Gap & Vertical Fault ............................................................... 22
Figure 2.6 .......... A Wedge, Gap & Sloping Fault ............................................................... 25
Figure 2.7 .......... Irregular Areal Grid .................................................................................... 28
Figure 3.1 .......... A filled reservoir grid .................................................................................. 35
Figure 3.2 .......... Interpolation Grid ....................................................................................... 40
Figure 3.3 .......... Block centred & node centred interpolation grids for a reservoir ............... 40
Figure 3.4 .......... Interpolation grid with data points & skeleton ........................................ 41
Figure 3.5 .......... Neighbors in the interpolation grid .......................................................... 42
Figure 3.6 .......... Interpolation grid after first pass .............................................................. 43
Figure 3.7 .......... Interpolation grid after second pass ......................................................... 43
Figure 3.8 .......... Flat Block Conversion to Corner Point Geometry .................................... 47
Figure 3.9 .......... Sloping Block Conversion to Corner Point Geometry ............................. 47
Figure 4.1 .......... Defining z co-ordinates of the grid block centres .................................... 54
Figure B.1 .......... Inverse distance interpolation along a straight line ............................... 127
List of Tables

Table 2.1  ZCORN data items .......................................................................................................................... 21
Table 2.2  COORD data items .......................................................................................................................... 23
Table 4.1  Keyword summary .......................................................................................................................... 57
FILL is designed to aid the engineer in preparing GRID Section data for input into ECLIPSE. The main purpose of FILL is to generate corner point data defining the shape and position of the grid blocks. Corner point geometry provides a highly flexible means of representing distorted and faulted grids. The corner point data required by ECLIPSE must specify the location of each corner of every grid block. This can be quite voluminous for large grids. FILL provides a means of generating this data from a much smaller set of keyword information. It contains a powerful set of techniques to represent vertical and sloping displacement faults.

Another use of FILL is in preparing the grid block property data, such as porosity, permeability etc. Using FILL, the engineer can set the grid block property values and can modify them selectively by addition, multiplication or substitution in specified sub-regions of the grid. These data editing operations could alternatively be performed directly in ECLIPSE, but FILL provides greater flexibility in setting the grid block properties. In particular, FILL can interpolate grid block property data supplied at selected locations in the grid, to estimate the corresponding values in the regions where the data is missing. This feature is useful for setting up reservoir data for a large field, using the data measured at the wells. Another example of the extra data editing flexibility in FILL is that one array can be divided by another array, which is useful for computing permeabilities from permeability-thickness and thickness data.

FILL produces an output file containing grid data that can be directly included in the ECLIPSE data input file. As this output can be quite voluminous, the user should not normally attempt to examine the contents of the file directly, but should preferably use the grid graphics facility in GRAF or the ECLIPSE output listing.

The grid graphics facility is strongly recommended as a means of checking the geometry of the grid, especially with complex faulted problems using corner point geometry. It may be used directly with the optional grid geometry file produced by FILL - there is no need to run ECLIPSE.
Defining the Reservoir Grid

Chapter 2

Introduction

ECLIPSE contains a wide range of geometry options enabling the engineer to construct a faithful representation of the reservoir geology with substantially fewer cells than would be required with conventional gridding systems. The grid may be distorted to represent the shape of the reservoir, and neighboring cells may be displaced to represent faults. The grid can be specified in this flexible manner by means of the Corner Point Geometry option. This section describes the concepts of corner point geometry, and illustrates how distorted and displaced grids can be set up with the aid of FILL. For a more complete description of the FILL keyword data, the user is referred to "The FILL Input File" on page 49 and the subsequent keyword descriptions.

Conventional simulators represent each grid cell as a regular cube or brick shape. Because bricks are often placed at varying depths to correspond to the shape of the reservoir, the data required for each brick are its dimensions and depth. This representation leads to conceptual difficulties for, although the pore volume is well defined, it is not always clear how the building bricks should be joined together and the definition of transmissibilities between neighboring cells is ambiguous. These difficulties become especially acute if the reservoir is faulted.

In ECLIPSE, the geometry problem is solved by defining each grid block by the location of its corner points. Thus ECLIPSE allows each cell to become highly distorted but, in the absence of faults, its corners will join smoothly with those of its neighbors. Transmissibilities and pore volumes are then well defined. Displacement faults can be defined by distorting cells to fit smoothly to the fault plane and by vertical displacement to preserve the bedding structure on either side of the fault. The resulting ‘non-neighbour’ connections are handled precisely by the ECLIPSE solution technique. A meaningful graphical representation of the grid, which is useful for data checking purposes, is also made possible by the corner point technique.

With corner point geometry, the pressures and saturations are still quantities relating to the grid blocks themselves, rather than the nodes representing the grid block corners. Thus in a mathematical sense, a corner point grid is still block-centred. However, we shall use the term ‘block-centred’ to refer to a grid specified in the traditional manner (dimensions and depth) rather than by using corner point geometry.
For completeness, grid data can be specified for ECLIPSE either in the traditional block-centred way or in the corner or node-centred way. Because ECLIPSE allows block-centred geometry to be specified quite generally, it is possible to use the traditional method to define a distorted grid; but there are limitations in doing so and the user could generate an impossible geometry which cannot be displayed graphically in a meaningful way. Faults can be specified most easily in corner point geometry.

Traditional grid systems can be specified directly for ECLIPSE using conventional keywords such as $DX$, $DY$, $DZ$, $TOPS$ or $DEPTH$. In corner point geometry the grid block shapes and positions are defined with the keywords $COORD$ and $ZCORN$. $COORD$ defines the position and slope of each ‘coordinate line’, on which the grid block corners are located, while $ZCORN$ defines the depth of all the grid block corners. The data for these two keywords can be quite voluminous, but FILL constructs it from a much simpler set of keyword information. FILL can also construct grid block property keyword data such as $PORO$, $PERMX$ etc. from a sparser set of data supplied as input, determining the missing values by interpolation. It is advisable to check the output from FILL using the ECLIPSE graphics package, to ensure that the correct grid system has been generated.

An important concept in the construction of grid data for corner point geometry is the idea of ‘co-ordinate lines’. Co-ordinate lines are straight lines upon which all the cell corners must lie. Thus if there are $NX$ cells in the X-direction and $NY$ cells in the Y-direction, there will be $(NX+1)*(NY+1)$ coordinate lines. In simple grids the co-ordinate lines are vertical, but in more general grids they can be off-vertical to coincide with sloping faults. Cell corners lie on coordinate lines just like beads on wires as illustrated in Figure 2.1. To define a grid it is sufficient to specify the co-ordinate lines and the depths of cell corners. The FILL program enables this to be done in a relatively simple way.

Figure 2.1 Co-ordinate lines & sloping faults
A simple conventional grid in ECLIPSE

A simple 4x4x4 grid (see Figure 2.2) is used below to clarify the ideas described above and to illustrate the input data required for ECLIPSE. Suppose this represents a reservoir which is 4000ft by 4000ft areally, 80ft thick and with the top of the reservoir at 5000ft. The conventional input grid data for ECLIPSE is

```
TOPS
  16*5000 48* /
DXV
  4*1000 /
DYV
  4*1000 /
DZ
  64*20 /
```

The TOPS keyword is followed by 16 items of data and 48 missing values, followed by a slash (/). This tells the program that the depths of the tops of the 16 cells in the top layer of the grid all have the value 5000ft. The missing values are used to pad out the list to the number of items expected for the TOPS keyword. The slash tells the computer to expect a new keyword to follow. The remaining data states that the dimensions of each of the 64 cells in the reservoir are 1000ft by 1000ft by 20ft.

**Note**

X and Y grid dimensions are always assumed to be measured horizontally in ECLIPSE and FILL. Z dimensions are assumed to be measured vertically downwards. Thus grid block dimensions are not measured along the bedding plane. However, X and Y permeabilities are assumed to be along the bedding plane direction. The Z permeability is assumed to be measured across the bedding plane.
A regular orthogonal grid can be specified in block-centered geometry or in node-centered geometry.

Centre-point geometry

Corner-point geometry
A conventional grid in FILL

One of the useful features of FILL is that it will ‘fill in’ missing values, given the relatively small amount of data vital to the specification of the grid. For example, certain items of data such as porosities, rock permeabilities, net to gross ratios, and, to a lesser extent, layer thicknesses are only known at wells. Using FILL, it is only necessary to specify data in selected cells such as well cells, and the program will then interpolate appropriate values for the remaining cells. Because rock properties such as permeabilities often vary dramatically from layer to layer, FILL interpolates properties only on a layer by layer basis and does not interpolate vertically. FILL can be used to specify either block-centred or node-centred geometry.

In block-centred geometry the data for FILL is similar to that for ECLIPSE:

```
SPECGRID
  4 4 4 1 F /
DXV
  4*1000 /
DYV
  4*1000 /
TOPS
  16*5000 48* /
DZ
  64*20 /
FILL
```

The SPECGRID keyword specifies the type of grid. In this case we have 4 blocks in the X-direction, 4 blocks in the Y-direction, 4 blocks in the Z-direction, 1 reservoir, and the radial option is false (F) which means that the grid is cartesian. The DXV and DYV keywords define an orthogonal grid system with four steps of 1000ft along the X-axis and four steps of 1000ft along the Y-axis. The step lengths do not all have to be the same size. For example, if the DXV keyword and its data were replaced by

```
DXV
  1000 2000 1500 500 /
```

we would generate an irregular, but nevertheless orthogonal, grid.

TOPS specifies the depths of the tops of the 16 cells in the top layer of the reservoir. The DZ keyword defines the thicknesses of all 64 cells to be 20ft. Finally, the keyword FILL is used to generate the grid data in a form suitable for ECLIPSE.
The simple grid in corner point geometry

To specify the same reservoir in corner point geometry, the data is

```
SPECGRID
  4 4 4 1 F /
DXV
  4*1000 /
DYV
  4*1000 /
TOPSNO2E
  25*5000 75* /
DZNODE
  100*20 /
FILL
```

Here the TOPSNO2E keyword states that the 25 nodes on the top of the reservoir are all located at a depth of 5000ft. DZNODE then locates the remaining 100 nodes each at 20ft vertically below the node above.
Filling in missing values

To illustrate how FILL interpolates missing values suppose the TOPSNODE data above is replaced by

```
TOPSNODE
5000 3* 5100 15* 5400 3* 5420 75* /
```

The data following the TOPSNODE keyword is interpreted as follows.

Node (1,1,1) is at 5000ft. The ‘3*’ followed by a ‘blank’ means that the next 3 values are ‘missing’. Thus no depths are given for nodes (2,1,1) to (4,1,1). Node (5,1,1) is at 5100ft. The next 15 values are ‘missing’. Node (1,5,1) is at 5400ft. The next 3 values are ‘missing’. Node (5,5,1) is at 5420ft.

Thus the depths of the nodes at the top of the reservoir are

```
IN = 1 2 3 4 5
1 5000 * * * 5100
2 * * * * *
JN = 3 * * * * *
4 * * * * *
5 5400 * * * 5420
```

The FILL operation fills in the ‘missing’ depths (those indicated by a *) and the depths of the top nodes become

```
IN = 1 2 3 4 5
1 5000 5025 5050 5075 5100
2 5100 5120 5140 5160 5180
JN = 3 5200 5215 5230 5245 5260
4 5300 5310 5320 5330 5340
5 5400 5405 5410 5415 5420
```

Missing values can also be input to the DZNODE keyword

```
DZNODE
20 2* 8 16* 20 2* 8 1*
20 2* 8 16* 20 2* 8 1*
20 2* 8 16* 20 2* 8 1*
20 2* 8 16* 20 2* 8 1* /
```

This states that the vertical thickness of each layer of the reservoir below each node is

```
IN = 1 2 3 4 5
1 20 * * 8 *
2 * * * * *
JN = 3 * * * * *
4 * * * * *
5 20 * * 8 *
```
which, after the **FILL** operation, becomes

<table>
<thead>
<tr>
<th>IN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>16</td>
<td>12</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>16</td>
<td>12</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>JN</td>
<td>3</td>
<td>20</td>
<td>16</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>20</td>
<td>16</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>20</td>
<td>16</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

**Note**  
FILL extrapolates beyond the specified range at a constant value. Note also that the data for each layer must be specified independently in `DZNODE` because, for physical reasons, the FILL interpolation procedure operates independently on each layer and does not act vertically. Thus at least one value must be given for each layer in `DZNODE`. The resulting grid is shown in **Figure 2.3**.

**Figure 2.3** A Wedge
A wedge or gap

**Figure 2.4** shows a non-contiguous reservoir with a wedge shaped inactive area which we wish to exclude. This might represent, for example, a shale bank containing immobile fluids. Before inserting the wedge we note that the depths of nodes at the top of the third layer in the previous contiguous example are

<table>
<thead>
<tr>
<th>IN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>JN</td>
<td>1</td>
<td>5040</td>
<td>5057</td>
<td>5074</td>
<td>5091</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5140</td>
<td>5152</td>
<td>5164</td>
<td>5176</td>
</tr>
</tbody>
</table>

which can be deduced by adding the **DZNODE** values for the two top layers to the depths of the nodes at the top of the reservoir. To specify the wedge these depths must be modified to

<table>
<thead>
<tr>
<th>IN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>JN</td>
<td>1</td>
<td>5060</td>
<td>5067</td>
<td>5074</td>
<td>5091</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5160</td>
<td>5162</td>
<td>5164</td>
<td>5176</td>
</tr>
</tbody>
</table>

**Figure 2.4** A Wedge & Gap
One way of achieving this result is to replace the `TOPSNODE` and `DZNODE` data sections with

```
TOPSNODE
 5000 3* 5100 15* 5400 3* 5420
 25*
 5060 5067 18* 5460 5447 3*
 25* /
FILL
TOPSNODE S
 ('=', 1*, 3 5, 1 5, 3 3) /
/
DZNODE
 20 2* 8 16* 20 2* 8 1*
 20 2* 8 16* 20 2* 8 1*
 20 2* 8 16* 20 2* 8 1*
 20 2* 8 16* 20 2* 8 1* /
FILL
```

The depths of the top nodes are the same as in the previous example. The depths of nodes at the top of the second layer are ‘missing’. After the first FILL operation the depths of nodes at the top of the third layer become

```
IN = 1  2  3  4  5
 1 5060 5067 5067 5067 5067
 2 5160 5162 5162 5162 5162
JN = 3 5260 5257 5257 5257 5257
 4 5360 5352 5352 5352 5352
 5 5460 5447 5447 5447 5447
```

At this stage all nodes with \( \text{IN} = 3, 4, 5 \) are incorrect. To understand the next step we need to know that FILL takes its depths from the layer above in preference to interpolation within a layer provided that the relevant `DZNODE` data is available. At the first FILL operation no `DZNODE` data had been read, so the depths of the third layer of nodes were filled out by interpolation (and constant extrapolation) of the sparse depth data given for that layer. We must therefore re-determine the node depths in layer 3 that were incorrectly extrapolated. To do so, we first of all make the 15 incorrect node depths ‘missing’, using

```
TOPSNODE
  S '='.$, 1*, 3 5, 1 5, 3 3) /
```

The format of the data following this keyword is different from the data formats we have used so far. By placing an ‘S qualifier’ after the keyword, we instruct the program to accept the subsequent data in ‘Single-value list format’. This assigns (‘=’) a “missing” value (1*) to all top nodes in the range \( \text{IN} = 3 \) \( \rightarrow 5 \), \( \text{JN} = 1 \) \( \rightarrow 5 \) and \( \text{KN} = 3 \) \( \rightarrow 3 \).

The depths of nodes at the top of the third layer are now

```
IN = 1  2  3  4  5
 1 5060 5067 * * *
 2 5160 5162 * * *
JN = 3 5260 5257 * * *
 4 5360 5352 * * *
 5 5460 5447 * * *
```

The final FILL operation following the `DZNODE` data fills in the desired depths for all nodes.
A vertical fault

We will now introduce a vertical fault into the reservoir using the keyword \texttt{ZCORN} which specifies the depths of grid block corners. Each grid block has eight corners and the data for \texttt{ZCORN} must state clearly which grid blocks and which corners are to be operated on. FILL keywords may be qualified by a letter following the keyword, separated by a space. An \texttt{A} indicates that an array of data will follow, while an \texttt{S} indicates that one or more lines containing a single value will follow. For example, a single depth adjustment may be applied to a number of nodes or corners. With an \texttt{S} qualifier, one or more lines of data follow the keyword. Table 2.1 shows the data item that each \texttt{ZCORN} line contains (in free form):.

Table 2.1 \texttt{ZCORN} data items

<table>
<thead>
<tr>
<th>Data item</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATOR</td>
<td>The operator enclosed in single quotation marks</td>
</tr>
<tr>
<td></td>
<td>‘=’ - assign or replace</td>
</tr>
<tr>
<td></td>
<td>‘+’ - add</td>
</tr>
<tr>
<td></td>
<td>‘*’ - multiply</td>
</tr>
<tr>
<td></td>
<td>‘-’ - subtract</td>
</tr>
<tr>
<td></td>
<td>‘/’ - divide</td>
</tr>
<tr>
<td>MODIFYER</td>
<td>The single value</td>
</tr>
<tr>
<td>ICRANGE</td>
<td>The corners in the I direction (1 - left, 2 - right) (e.g. 1 2 = 2* = both corners, 1 1 = left corner only)</td>
</tr>
<tr>
<td>IBRANGE</td>
<td>The first and last grid block numbers in the I direction</td>
</tr>
<tr>
<td>JCRANGE</td>
<td>The corners in the J direction (1 - back, 2 - front)</td>
</tr>
<tr>
<td>JBRANGE</td>
<td>The first and last grid block numbers in the J direction</td>
</tr>
<tr>
<td>KCRANGE</td>
<td>The corners in the K direction (1 - top, 2 - bottom)</td>
</tr>
<tr>
<td>KBRANGE</td>
<td>The first and last grid block numbers in the K direction</td>
</tr>
</tbody>
</table>

For example, to introduce a fault by lowering all grid blocks with \texttt{I = 4} and \texttt{J = 2, 3 and 4} by a distance of 10ft, we would append the following data:

\texttt{ZCORN S}

\texttt{‘+’ 10.0, 1 2, 4 4, 1 2, 2 4, 1 2, 1 4 /}

The resulting reservoir is illustrated in Figure 2.5.
Figure 2.5  A Wedge, Gap & Vertical Fault
In the previous example we showed how to define a vertical fault. To specify a sloping fault we must first introduce the coordinate lines along which the fault may be defined. Normally coordinate lines are vertical lines drawn through nodes with the same (IN,JN) indices. In general, however, coordinate lines do not have to be vertical and, using the `COORD` keyword, it is possible to define sloping coordinate lines. Given the depth of a particular grid block corner, and the associated coordinate line, the X and Y coordinates of the corner can be calculated by ECLIPSE. In `S` format, the data items required for each line after `COORD` are

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATOR</td>
<td>The operator ‘=’ assign ‘+’ add etc.</td>
</tr>
<tr>
<td>MODIFIER</td>
<td>The single value</td>
</tr>
<tr>
<td>DIMENSIONS</td>
<td>The first and last dimension</td>
</tr>
<tr>
<td>1 - X</td>
<td></td>
</tr>
<tr>
<td>2 - Y</td>
<td>e.g. 1 3 means X to Z i.e. X, Y and Z</td>
</tr>
<tr>
<td>3 - Z</td>
<td></td>
</tr>
<tr>
<td>POINTS</td>
<td>The point on a line</td>
</tr>
<tr>
<td>1 - top</td>
<td></td>
</tr>
<tr>
<td>2 - bottom</td>
<td>e.g. 1 2 means both top and bottom points on a line</td>
</tr>
<tr>
<td>INRANGE</td>
<td>The grid node range in the I direction</td>
</tr>
<tr>
<td>e.g. 1 7</td>
<td>means all grid nodes from IN = 1 to IN = 7</td>
</tr>
<tr>
<td>JNRANGE</td>
<td>The grid node range in the J direction e.g. 3 3 means only grid nodes with JN = 3</td>
</tr>
<tr>
<td>RESNUMBER</td>
<td>The reservoir number (for grid systems with more than one reservoir)</td>
</tr>
</tbody>
</table>

Thus

```
COORD S
  "+" 50, 1 1 , 2 2 , 4 4 , 3 5 /
/
```

adds 50ft in the X-direction to the bottom of all coordinate lines with node coordinates IN = 4 and JN = 3, 4, 5.
**Figure 2.6** shows a reservoir containing a sloping fault generated by the following commented data

```plaintext
-- Reservoir number 1 has 4 blocks in the x-direction, 4 blocks in the y-direction and 4 blocks in the z-direction. The radial option is set to FALSE so a cartesian grid is specified.
-- The SPECGRID format is
SPECGRID
4 4 4 1 F /
-- The regular areal grid has 4 1000 ft blocks in both the x and y directions.
-- This defines x and y for the coordinate lines.
DXV
4*1000 /
DYV
4*1000 /
-- Specify tops of top layer and part of third layer.
TOPSNODE
5000 3* 5100 15* 5400 3* 5420 25*
5060 5067 18* 5460 5447 3* 25* /
-- Fill out the depths of nodes in the top and third layers.
-- Set the z coordinates for the top and bottom of the coordinate lines to the node depths of the top layer.
FILL
-- Make missing the tops of the third layer of blocks which must be calculated from DZNODE data; as depths are taken from the layer above in preference to interpolating. The relevant blocks have
-- node values in the range
-- IN = 3 TO 5, JN = 1 TO 5, KN = 3 TO 3.
-- The TOPSNODE S format is
TOPSNODE S
'=', 1* , 3 5 , 1 5 , 3 3 / /
-- Specify the reservoir thickness at nodes.
DZNODE
20 2* 8 16* 20 2* 8 1* 
20 2* 8 16* 20 2* 8 1* 
20 2* 8 16* 20 2* 8 1* /
```
--- Depress the fault region by a vertical distance of 10 ft.
--- The block range is IB = 4, JB = 2 TO 4, all KB.
--- ZCORN S format
--- OP VALUE L/R IBRANGE B/F JBRANGE T/B KBRANGE
ZCORN S
  '+' ,10.0, 2* , 4 4 , 2* , 2 4 , 2* , 2* /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /

--- At this stage the depths of tops and bottoms of coordinate lines
--- have been defaulted to the top of the reservoir.
--- To introduce a 30 degree sloping fault in the x-direction we first
--- redefine the bottoms of coordinate lines at 173.2 ft below the top
--- and move them 100 ft in the x-direction.
--- The relevant coordinate lines are in the range IN = 4 and JN = 2 TO 5
--- The COORD S format is
--- OP DIST DIR T/B INRANGE JNRANGE
COORD S
  '+' 173.2 3 3 2 2 /

COORD S
  '+' 100.0 1 1 2 2 4 4 2 5 /
Interpolating rock data

Rock properties such as porosities and permeabilities can also be interpolated on a layer by layer basis using FILL. Editing however can be carried out for several layers at a time. For example

```
PORO
16*0.2
0.22 2* 0.16 8* 0.1 2* 0.31
16*0.25
16*0.1 /
FILL
```

results in the following porosity distribution on the second layer

```
IB = 1  2  3  4
1 0.22 0.20 0.18 0.16
2 0.18 0.19 0.20 0.21
JB = 3 0.14 0.18 0.22 0.26
4 0.10 0.17 0.24 0.31
```

For irregular grid systems, property interpolations take account of grid block dimensions and locations. To edit this data we might put

```
-- OP MODIFIER IBRANGE JBRANGE KBRANGE
PORO S
'+' 0.1 2 3 2 3 2 3 /
'**' 0.9 2 3 2 3 2 3 /
/
```

giving

```
IB = 1  2  3  4
1 0.22 0.20 0.18 0.16
2 0.18 0.261 0.27 0.21
JB = 3 0.14 0.252 0.288 0.26
4 0.10 0.17 0.24 0.31
```
A distorted areal grid

Using corner point geometry it is possible to define highly distorted grid systems which may conveniently represent the shape of the reservoir with relatively few cells. Large cells should be placed in the aquifer region with small cells near wells. The edges of distorted cells should be aligned along faults and, where possible, aligned also in the direction of expected fluid flow.

Figure 2.7 shows a simple distorted grid. The data required to generate this grid given below, with explanatory comments:

```
SPECGRID
  4 4 4 1 F /
  -- The COORD keyword is used in array format to specify pairs of
  -- x and y coordinates for the tops of coordinate lines.
  -- OPERATOR DIMENSIONS TOP/BOTTOM
  -- SET X,Y TOP ONLY
COORD A
  '=', 1,2, 1,1 /
   1 1 , 2 0 , 4 0 , 6 0 , 6.5 0.5
   0 2 , 3 3 , 4 3 , 5 3 , 7 2
   1 4 , 3 4 , 4 4 , 5 4 , 7.5 4
   1 5 , 3 5 , 4 5 , 5 5 , 8 6
   1 6 , 3 6 , 4 7 , 6 8 , 8 9 /
  -- A scale factor may now be used to scale up to field dimensions.
COORD S
  '***' 1000.0 /
/
  -- Depths of the nodes at the top of the reservoir.
TOPSNODE
  25*5000 75* /
  -- All four layers have the same thickness.
DZNODE
  100*20 /
  -- Use FILL to compute the node depths.
FILL
```
Figure 2.7 Irregular Areal Grid
Geometry conversion

FILL accepts grid block depth and thickness data in block-centred form (i.e. with keywords TOPS, Z and DZ), and converts them into corner point form (output under keyword ZCORN). This process is referred to as “geometry conversion”. There is a choice of two methods for geometry conversion.

The first method is the flat-block method, selected by the FLATBLCK keyword. This results in blocks with flat (horizontal) top and bottom faces. The four corner depths at the top of a block, for example, will all be equal to the specified or interpolated TOPS value for the block. If the top surface of the neighboring block has a different depth, there will be a step change in depth at the interface between the two blocks.

The flat-block method, therefore, results in a grid in which the top and bottom surfaces of neighboring grid blocks are discontinuous. Whenever this method is used, the OLDTRAN switch should be used in the corresponding ECLIPSE run (see the "ECLIPSE Reference Manual"). If the NEWTRAN switch is used instead, ECLIPSE will interpret these depth discontinuities as actual faults, producing a vast number of non-neighbour connections.

The flat-block method has been provided to enable FILL to reproduce the grid geometry that would result from the direct use of the keywords TOPS and DZ in ECLIPSE. It is the default conversion method in FILL, i.e. it will be selected if neither the FLATBLCK nor the SLOPBLCK keywords are specified.

The second method is the sloping-block method, selected by the SLOPBLCK keyword. This results in blocks with sloping top and bottom faces that are continuous from one block to the next. The four corner depths at the top of a block, for example, will be interpolated from the filled-in TOPS values of the adjacent set of blocks. The top centre depths calculated from these corner depths by ECLIPSE will not be exactly the same as the originally specified (or interpolated) TOPS values, due to the smoothing involved in the interpolation process.

The method uses constant value extrapolation. Thus the top outer corners of a layer will have the same depths as the original top-centre depths of the outer blocks.

The NEWTRAN switch in ECLIPSE can be used with this method. It will give more accurate transmissibilities, and it will not produce any faults unless they have been specifically introduced by the user.

These methods are discussed in more detail in the section entitled "A detailed description" on page 38.
Multiple reservoirs

The term “reservoir” has a special meaning in the FILL program. Each reservoir has its own coordinate system, including its own set of coordinate lines. The grid blocks in a reservoir are specified by a range of values of the k index: thus a reservoir occupies a distinct “box” of (i, j, k) values. Two reservoirs cannot have any grid blocks in common. The only connection between reservoirs should be via wells and non-neighbour connections.

For normal problems only one reservoir is required. Typical applications for multiple reservoirs are:

- where two reservoirs at different depths require different (x, y) scales for the grid blocks.
- where two reservoirs at different depths have faults with greatly different slopes.
- local grid refinement, where a fine-grid system of small grid blocks is substituted for a single grid block in a coarse-grid system. The fine-grid system is set up as a separate reservoir, and is connected to the coarse grid blocks by non-neighbour connections.

The user must ensure that the z transmissibilities between blocks in adjacent reservoirs are set to zero, or specified explicitly. It is best to set up connections between such blocks using non-neighbour connections.
Missing values

A “missing value” for a data item means that the data item is empty, and is to be filled-in by the FILL program using interpolation or other means (e.g. by adding dz’s onto depths). The filling-in is carried out during the FILL operation, signalled by the FILL keyword.

A single missing value is represented by “1*” in free-format input. In this manual it is represented as <missing>. If the user wishes to set all values to <missing>, then this should be indicated explicitly, e.g. as “120*”. It is an error to specify less data for a keyword than required by that keyword, as indicated by a slash (/) terminating the input prematurely. When this error occurs, the remaining items are assigned missing values.
Input to ECLIPSE

Output from FILL may be input directly into ECLIPSE. The most convenient way of doing this is to ‘include’ the output file from FILL in the GRID data section of the input data for ECLIPSE. For example, suppose the output file from FILL is named FILLED, then the following section of data should be inserted in the ECLIPSE input file:

```
INCLUDE 'FILLED' /
```

FILL generates data under keywords recognized by ECLIPSE. For example, grid geometry data is generated under the keywords COORD and ZCORN. COORD specifies coordinate lines and ZCORN specifies the depths of each grid block corner. Since each grid block has 8 corners there are 8 items of data for each grid block under ZCORN. There are, of course, 6 items of data specifying each coordinate line (the X, Y, Z or R, THETA, Z for two points on each coordinate line). This large quantity of data defines the shape of the reservoir for ECLIPSE, no matter how complex it may be. It also enables the graphics program to be used to draw pictures of the reservoir to help ensure that the data represents the shape of the reservoir correctly. The COORD and ZCORN keywords are not intended for direct user input of grid data for ECLIPSE other than via FILL, although they can be used in this way.

If sufficient data has not been provided to define the grid fully, or to determine all the missing data values, then the grid blocks with undetermined values will be made inactive by setting their active cell indices to zero in the ACTNUM keyword array. The ACTNUM keyword, which identifies the active grid blocks, is included by default in the output from FILL. Any data values that remain undetermined will appear as $-1.0E20$ in the output of their respective keyword arrays.

**Note** ECLIPSE may make a grid block inactive, even if its ACTNUM value is 1, if its pore volume is zero.

The user can suppress the output of any unwanted keywords by including the OUTFILL keyword in the input. For example, this can be used to suppress the output of the ACTNUM keyword. Another instance where this would be useful is when FILL is being used solely for interpolating grid block property data such as PORO, PERMX etc.; OUTFILL could then be used to suppress the output of all the unwanted keywords.

This section has outlined the capabilities of the Corner Point Geometry option, and has given examples of how a complex grid can be generated with FILL. For a detailed discussion of the input data requirements, the user is referred to the section entitled “The FILL Input File” on page 49 and the subsequent keyword descriptions.
Notes

There are several points to keep in mind when using FILL.

1. The user needs to be aware of the order in which variables are processed during the FILL operation. For instance, if the depth at a bottom corner of a grid block has a non-missing value at the start of the FILL operation, then that value will not be modified - even if the depth is available at the corresponding top corner, and the thickness is available. This situation can arise if a single depth is supplied for the bottom of a layer of blocks, and the FILL keyword is then used, so filling the bottom of the layer with that depth value (in the absence of thickness data in that layer).

2. For any block-centred variable, such as porosity, if there is a single data value in a layer of blocks and the FILL operation is applied: then the layer will be completely filled with that constant value by the interpolation procedure. The interpolation procedure has the same behavior for layers of corners in ZCORN, and so on; however other depth values may be introduced before interpolation, as described below.

3. Depths are processed by layers, from top to bottom. Before interpolating the top corners of a layer, depths are transferred from the layer above - but only where they are missing in the top of the current layer.

4. After interpolating the top corners of a layer, the thicknesses are interpolated. Then, where a corner depth in the bottom of a layer is missing, it is calculated using the corresponding top depth and thickness (provided these are available).

5. If no depths have been specified before doing a FILL, but some thicknesses have been specified, the thicknesses will be filled out but none of the depths will be affected. This can be useful when several fills are necessary for setting up the thicknesses, to avoid the depths being prematurely filled out with the wrong values.

6. Suppose a layer of permeability data, PERMI, has been completely filled in, perhaps with a constant linear trend. Suppose the user wants to modify a sub-area of the layer using interpolation, as indicated by *'s below, to establish another trend, independent of the original data.

```
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
- - - - - - * * * * * * - - - -
- - - - - - * * * * * * * * - - - -
- - - - - - * * * * * * * * * * - - - -
- - - - - - * * * * * * * * * * * * - - - -
- - - - - - * * * * * * * * * * * * * * - - - -
- - - - - - * * * * * * * * * * * * * * * * - - - -
- - - - - - * * * * * * * * * * * * * * * * * * - - - -
- - - - - - * * * * * * * * * * * * * * * * * * * * - - - -
```

This can be done as follows. Firstly, set the values of the * points to <missing>, by setting them to “n* “ using any combination of A and S formats. Then give non-missing values to all the boundary points of the sub-area, to prevent information from the surrounding area “leaking in”. These are indicated by X’s below.
The **FILL** may now be performed. All the sub-area data may be set up at the same time: for instance one set of statements for setting up this data (where appropriate values should be given to the X's) is

```
SPECGRID
17 8 1 /
PERMI A
 8 8 2 2 1 1 /
  x /
PERMI A
 6 11 3 3 1 1 /
  x 1*  x  x /
PERMI A
 4 12 4 4 1 1 /
  x  x 6*  x /
PERMI A
 4 11 5 5 1 1 /
  x 6*  x /
PERMI A
 7 8 6 6 1 1 /
  x  x /
```

7 Indiscriminate use of the **FILL** keyword may have undesired effects, some of which have already been indicated, but **FILL** will remember which variables have already been filled or input. In case of difficulty, the progress of the calculations may be logged using the switches in the **OUTPUTS** keyword.

8 When using multiple reservoirs, note that the **FILL** operation will not transfer any depth information from one reservoir to another. Within a single reservoir, the bottom depths of one layer may be transferred to the top of the layer below, depending on what the user has specified. - When using faults, it is convenient to set up and verify the unfaulted depths, using the keywords **TOPSNODE** and **DZNODE**, with **FILL**. Then the faults may be added using **DZCORN** and **ZCORN**.

9 When performing multiple fills and modifications on **DZ**, **DZNODE**, and **DZCORN**, all the thickness data should be placed before the depth data (**Z**, **TOPS**, **ZNODE**, **TOPSNODE** and **ZCORN**). This makes sure that the depths remain missing until the thicknesses are frozen at their final values. Alternatively, any depths which have been filled in using obsolete thickness data should be made missing before the final **FILL** is performed.
Introduction

This chapter describes in detail what the FILL program does, and explains the algorithms used.

Figure 3.1 A filled reservoir grid
Outline of the program

Before proceeding with the discussion of the algorithms, it is important to understand the way in which missing values are handled.

Missing values

A “missing value” for a data item means that the data item is empty, and is to be filled in by the FILL program using interpolation or other means. This filling in is carried out during the FILL operation, signalled by the FILL keyword.

The FILL program initializes all variables except ACTNUM to <missing> at the beginning. At the end of the program each grid variable written to the output file is examined for missing values. If any active grid block has a missing value, that block is made inactive by setting ACTNUM to zero, so that ECLIPSE will not misinterpret the missing values.

Overview

FILL interprets every keyword (and its data) as it is read from the input file. This means that the data output by FILL may depend on the order in which data appears in the input. There is no specific order required by FILL, except that SPECGRID must come first to define the grid dimensions.

There is a large workspace in FILL. When the SPECGRID keyword has been read and the dimensions verified, every grid variable is allocated room in the workspace. If there is insufficient room, then an error message is given and no further input is read - FILL must be recompiled with a larger workspace.

Several variables are not allocated separate space, but share common space:

- XV are stored in COORDYVRADVTETAV
- DTHETAV is stored in DYV
- TOPSNODE are stored in ZCORNZNODE
- DZNODE is stored in DZCORN

Following the allocation of storage, all grid variables except ACTNUM are initialized to <missing>. ACTNUM is initialized to 1 - all grid blocks are active to begin with.

Reading a grid keyword

When a grid keyword (such as TOPSNODE) is encountered, all the data associated with that keyword are read into a temporary area. There are three basic data formats in FILL:

- ARRAY format,
- ECLIPSE format and
- SINGLE value modifiers format.

These are described in Appendix A.
Considering the **ARRAY** format, the index bounds indicate which elements (blocks, nodes etc.) of the variable are to be modified. The new data supplied is used to modify the old data in the workspace, but only in the region specified by the index bounds. The operator ("=", "+" etc.) determines how the old data is modified by the new data.

If the operator is "=" , then the new data will replace the old data, with each new value overwriting the corresponding old value(s). (Note that each **TOPSNODE** element is associated with 4 **ZCORN** values, namely the corner depths at the top of the adjacent blocks.) Any <missing> value in the new data will overwrite the old value(s), making it (them) <missing>.

For the other operators, the old data will be replaced by

\[
\text{old value} \ <\text{operator}> \ <\text{new value}>
\]

for example

\[
\text{old value} \ / \ <\text{new value}>
\]

If a new value is <missing>, or for some other reason the operation cannot be performed (for example the new value is zero for division), then the old value is left unchanged.

**Note** Only the "=" operator is allowed for integer data.

For the single-value format, each line of data in the list is treated in much the same way as for the array case. The difference is that only a single value is supplied to modify all the old data in the bounded region, for that variable.

Subsequent keywords with the same storage variable in the workspace will continue to modify the values of the variable.

**Reading other keywords**

When the **FILL** keyword is encountered, the **FILL** operation is performed on the workspace. This is described below. When keywords for switches are encountered (e.g. **FLATBLCK** and **SLOPBLCK**), the internal switches are modified accordingly. When **END** is read or the end of the input file is reached, all variables which have been selected for output are processed (see "**OUTFILL** on page 88). Each such variable is scanned for missing values. Any grid block with missing values is made inactive by setting **ACTNUM** to zero. Finally, the variables are written.
A detailed description

When the FILL keyword is encountered, the program will examine all the variables and attempt to fill in any missing values, by means of data copying and interpolation algorithms. This section contains a detailed description of these algorithms. For simple problems the user need not be concerned with these details, however when complex geometry is used, some knowledge of the ordering of operations is essential. In most cases the order should be what the user would expect; some points to note are listed at the end of the section on defining the grid.

Sequence of operations

1. The COORDSYS data is verified. If invalid, the FILL operation is ended.
2. The x and y coordinates of coordinate lines are filled as follows, using DXV, DYV and COORD, looping over reservoirs. For each layer of points (top or bottom) on coordinate lines in a reservoir, the coordinate lines are defaulted from the layer above, if necessary. For the current layer, the x coordinate is filled in two passes over the rows. (JN is constant for a row.) Dx’s are added to x’s where necessary: if an x value is not missing, then the dx will not be used to replace that x value with a calculated value. The first pass is backwards - from front to back over rows (decreasing JN), and within each row from right to left (decreasing IN):
   a. For a given row, if dx is not missing for column 1 (i = 1), and x is missing at nodes 1 and 2, then x(1) is defaulted from x(1) in the previous row (larger JN).
      The second pass is forwards - from back to front over rows (increasing JN), and within each row from left to right (increasing IN):
   b. If the right-hand x is missing, and either dx or the left-hand x is missing, then the right-hand x is defaulted:- from the previous row (smaller JN), where possible;- to zero, for the first row for node 1.
      This puts the coordinate origin at zero, if no x value is specified. The y coordinate is treated similarly.
3. Node-centred interpolation grids are constructed for each reservoir, using the (x,y) coordinates of the top point on each coordinate line. Since some of these values may be missing, a dummy interpolation grid is set up using the node indices, with
   
   dummy x = in
   dummy y = jn,

   and the Basic Interpolation Algorithm is applied twice for each reservoir to fill in the missing x and y values, if possible. COORD remains unchanged. If any node-centred interpolation grids are incomplete at the end of this step, the FILL operation is ended.
4. Block-centred interpolation grids are constructed from the node-centred ones: by taking the mean of the 4 points surrounding a block-centre. This ignores any z variation.
5. All the real block-centred variables, except TOPS, DZ and Z, are filled. The fill is done from top to bottom, by reservoir and by layer, using the block-centred interpolation grids and the Basic Interpolation Algorithm.
6 Any TOPS, DZ and Z are converted to node-centred form, and any missing values in ZCORN and DZCORN are replaced by values from this node-centred array. This procedure is also by reservoirs and by layers, from top to bottom.
   a Firstly Z and DZ are filled for the layer, using the block-centred interpolation grids. At points where TOPS is missing, values are calculated from Z and DZ where possible.
   b TOPS is then filled.
   c TOPS is converted to ZCORN form using the Block-centred to Node-centred Conversion Algorithm, described below.
   d DZ is then converted to DZCORN by the same method.
   e The next layer repeats this procedure.

7 The z and dz data in ZCORN and DZCORN is combined and stored in ZCORN, and COORD z values are defaulted.
   a Starting with the top layer of blocks, any depths from the bottom of the layer above (not the first layer) are used to fill in missing values in the top of the current layer.
   b Now the z’s of the top corners of the current layer are filled using the Basic Block-Corner Fill Algorithm (see below).
   c Similarly the dz’s are filled.
   d The z’s and dz’s are combined to fill in missing values for the z’s of the bottom corners of the current layer.
   e The bottom corners are filled.
   f Note that no data is transferred from one reservoir to the next.

---

**Interpolation**

With real grid data, such as porosities and depths, interpolation and extrapolation may be used to advantage to fill in the <missing> values. This means that the user need only specify enough data to describe the variable, so that the interpolation procedure can interpolate reasonable values for the rest.

Interpolation requires some geometrical information, namely the locations of all points where data is available and where data is wanted. In FILL all interpolation is performed within layers. The geometric information used is a two dimensional array of (x,y) values, called an “interpolation grid” (see Figure 3.2). The interpolation grid is always drawn as rectangular: it is not a scale drawing of the (x,y) locations of the grid points. No x or y value in the interpolation grid may be missing. (This is taken care of during the FILL operation: suitable (x,y) values are supplied, where missing, by a simple interpolation procedure.)
With block-centred data, such as porosity, an \((x,y)\) value must be provided for each block-centre \((i,j)\) location in each reservoir. With node-centred grid data, such as depths of grid block corners, an \((x,y)\) value must be provided for each \((i,j)\) node location in each reservoir.

One node-centred and one block-centred interpolation grid are used for each reservoir (see Figure 3.3). No account is taken in the interpolation procedure of the variation of \((x,y)\) position with \(z\), in the case of sloping coordinate lines. Also no account is taken of variation of \((x,y)\) across faults: the average values at the nodes are used. The interpolation grids are derived from \textsc{COORD}, as described in the next section.

Given an interpolation grid, and at least one data value defined at the grid points, the Basic Interpolation Procedure will fill in values for all points of the grid. Note that the \textless{} missing\textgreater{} value indicates which points do not have data. The algorithm uses inverse distance weighted averages of the values at the “neighbors”. The neighbors are the next grid points in the \(i^+\)- and \(j^+\)- directions with non-missing values: that is, along the lines of the interpolation grid. Each grid point has at most 4 such neighbors. (See Figure 3.5.)
All interpolated values will lie within the range of the original data points. All extrapolation is constant extrapolation.

### Basic interpolation algorithm

This algorithm fills in missing values for data points on an interpolation grid. The fill is performed in three passes, and is based on the “skeleton” of the data points. In each pass, all new values are placed in a work array and the data array is not updated until the end of the pass.

The “skeleton” is the set of i lines and j lines containing at least one non-missing data value. An “i line” is an interpolation grid line with constant i, varying j. Thus every data point is an intersection point of the skeleton (See Figure 3.4).

![Figure 3.4](image)

“Nearest” here does not refer to cartesian distance, but means the next i or j, along a grid line, with a non-missing data value.

1. The skeleton is determined.
2. All intersection points of the i lines and the j lines of the skeleton are filled, as follows.
   a. Where an intersection point already has a data value, this value is used.
   b. Otherwise, an average is taken of the (at most 4) nearest non-missing neighbors in the + and - i and j directions, by applying the Basic Averaging Algorithm (see Figure 3.5).
3. Points on either an i line or a j line of the skeleton are filled.
   a. Only the (at most) 2 nearest neighbors along the lines of the skeleton are used in each case, and the Basic Averaging Algorithm is applied to those points.
4. Points not on the skeleton are filled.
   a. The Basic Averaging Algorithm is applied to the (at most) 4 nearest neighbors in the + and - i and j directions. Note that such neighbors must be in the skeleton.

It will be seen in the following example that with this technique, extrapolation is performed by repeating the last value, i.e. constant-value extrapolation rather than constant-gradient extrapolation. Also, under certain conditions the interpolation method reduces to bilinear interpolation.
**Example**

Consider the layer of porosity data in the section entitled "Interpolation of porosity (1)" on page 132. During the FILL process, this will be loaded into a 5 x 4 interpolation grid. In this example the coordinate system is very regular - the (x,y) reservoir grid is made up of 1000 ft x 1000 ft squares, so the distances are simple to work out. Initially the interpolation grid looks like

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1*</td>
<td>0.1</td>
<td>1*</td>
<td>1*</td>
<td>0.2</td>
</tr>
</tbody>
</table>
| 1*| 1*| 1*| 0.3| 1*
| 1*| 1*| 1*| 1*| 1*|
| 1*| 1*| 0.4| 1*| 0.05|

where “1*” represents a missing value. See also Figure 3.4 to Figure 3.7.

After the first pass of the Basic Interpolation Algorithm, with the intersection points filled in, the grid looks like

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1*</td>
<td>0.1</td>
<td>0.1818</td>
<td>0.22</td>
<td>0.2</td>
</tr>
<tr>
<td>1*</td>
<td>0.1667</td>
<td>0.3333</td>
<td>0.3</td>
<td>0.21</td>
</tr>
<tr>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
<td>1*</td>
</tr>
<tr>
<td>1*</td>
<td>0.325</td>
<td>0.4</td>
<td>0.24</td>
<td>0.05</td>
</tr>
</tbody>
</table>
The intersection points are calculated using the Basic Averaging Algorithm, discussed below. For instance, the value of 0.1818 at (3,1) is obtained from the three points 0.1 \* 1* \* 0.2 by the inverse distance weighted average

\[
v = \frac{\left( \frac{1}{1000} \right) 0.1 + \left( \frac{1}{2000} \right) 0.2 + \left( \frac{1}{3000} \right) 0.4}{\left( \frac{1}{1000} \right) + \left( \frac{1}{2000} \right) + \left( \frac{1}{3000} \right)} = 0.1818
\] [EQ 3.1]

The other points are obtained similarly. After the second pass the interpolation grid has its skeleton filled in completely, and looks like

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.1667</th>
<th>0.325</th>
<th>0.1818</th>
<th>0.22</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.3333</td>
<td>0.3767</td>
<td>0.22</td>
<td>0.21</td>
<td>0.4</td>
</tr>
<tr>
<td>1*</td>
<td>0.2458</td>
<td>0.3667</td>
<td>0.27</td>
<td>0.13</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>0.325</td>
<td>0.325</td>
<td>0.4</td>
<td>0.24</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For instance, the value of 0.27 at (4,2) is obtained from the two points 0.31 \* 0.24 by the inverse distance weighted average
After the third pass the interpolation grid is filled completely, and looks like

\[
\begin{align*}
    v &= \frac{\left(\frac{1}{1000}\right)0.3 + \left(\frac{1}{1000}\right)0.24}{\left(\frac{1}{1000}\right) + \left(\frac{1}{1000}\right)} = 0.27
\end{align*}
\]  

[EQ 3.2]

as shown in the output file of this problem.

**Basic averaging algorithm**

The data value for a point is calculated from the data values for a list of neighboring points, given the (x,y) coordinates of all the points.

- If no neighboring points have the same x and y coordinates as the point being considered, the interpolated value is taken to be the weighted average of the neighboring values, using inverse distance weighting. This will be the normal case.
- Otherwise the interpolated value is taken as the unweighted average of the values of those neighboring points having the same x and y coordinates as the point being considered. This case can arise with inactive grid blocks (where the grid may not have been fully defined), or when a block has an x or y edge with zero length.

The resulting interpolated value will always lie within the range of the original data values.

**Basic block-corner fill algorithm**

This algorithm fills in data for the corners in a single layer of nodes, using a node-centred interpolation grid. There are four “node-corners” surrounding each node in the layer.

1. The block-corner data is loaded into a node-corner work array.
2. The corners at each node are filled, using only the information at the node.
   a. Each corner of the node is examined, and if missing a new value is calculated as the average of the adjacent old values for the node, if at least one adjacent value exists. Otherwise the opposite value is used as the new value. This is illustrated below, case by case. When all 4 corners have been examined, the new values replace the old values.
   b. All node-corners are now filled by applying the Basic Interpolation Algorithm 4 times: once for each type of corner, and each with its own work array.
3. The node-corner data is stored into the block-corner array.

**Node-corner filling**

(By cases)
Case 1
No values at the node.

Case 2
A single value “a” at the node.

Case 3
Two adjacent values “a” and “b” at the node.

Case 4
Two opposite values “a” and “b” at the node.

Case 5
Three values “a”, “b” and “c” at the node.

where \( d = \text{mean}(b,c) \)
Case 6

Three values at the node, with two identical adjacent values “a” and “b”. This is a special case of case 5.

\[
\begin{array}{c|c}
| & a \\
\hline
 b & a \\
\end{array}
\]

becomes

\[
\begin{array}{c|c|c}
| & a & a \\
\hline
 b & c & b \\
\end{array}
\]

where \( c = \text{mean}(a,b) \).

Case 7

Three values at the node, with two identical opposite values “a” and “b”. This is also a special case of case 5.

\[
\begin{array}{c|c}
| & a \\
\hline
 b & b \\
\end{array}
\]

becomes

\[
\begin{array}{c|c|c}
| & a & b \\
\hline
 b & b \\
\end{array}
\]

Case 8

Four distinct values “a”, “b”, “c” and “d” at the node.

\[
\begin{array}{c|c}
| & a \\
\hline
 c & b \\
\end{array}
\]

becomes

\[
\begin{array}{c|c|c}
| & a & b \\
\hline
 c & d \\
\end{array}
\]

Block-centred to node-centred conversion algorithm

This algorithm converts block-centred data for a layer of nodes to corner point form, using a node-centred and a block-centred interpolation grid. There are two methods (see “Conversion of block-centre depths (TOPS, DEPTH, DZ) (5)” on page 137.)

Flat-block method

For each \((i,j)\) block coordinates, set the 4 corner values to the block-centre value. (See Figure 3.8.)
**Sloping-block method**

This method ensures that the resulting corner data is continuous at nodes - i.e. the 4 values at a node are identical. (See Figure 3.9.)

1. Firstly a large interpolation grid is set up of size $(2*nx + 1) \times (2*ny + 1)$

   a. Both the block-centred and the node-centred interpolation grids are loaded into it, and then the other $(x,y)$ coordinates along odd interpolation grid lines are calculated as the mean of the two adjacent node-centred points.
2 The block-centred data is loaded and the Basic Interpolation Algorithm is applied.
3 The relevant data is stored into the corner array.

**Note** For the top of a layer, the depths of the edge nodes will be the same as the depths of the edge blocks, due to the constant extrapolation feature of the Basic Interpolation Algorithm.
The input to FILL consists of a sequence of keywords, each followed by its associated data. The keywords fall into three main categories:

1. Geometry keywords, which specify the grid block shapes, sizes and locations,
   - COORD, DEPTH, DTETAV, DXV, DYV, DZCORN, RADV, THETAV, TOPS, TOPSNOE, XV, YV, Z, ZCORN, ZNODE
   - together with
   - COORDSYS, SPECGRID and MAPAXES
   - which provide certain information about the grid as a whole.

2. Grid block property keywords, which define the grid block properties such as porosity and permeability,
   - ACTNUM, DEPTH, MULTI, MULTJ, MULTK, MULTTHT, MULTX, MULTY, MULTZ, NTG, PERMI, PERMJ, PERMK, PERMR, PERMTHT, PERMX, PERMY, PERMZ, PORO, PORV, TRANI, TRANJ, TRANK, TRANR, TRANTHT, TRANX, TRANY, TRANZ

3. Control keywords, which act as controls and switches to govern the operation of FILL,
   - DEBUG, END, FILL, FLATBLCK, INCLUDE, OUTFILL, OUTPUTS, SLOPBLC.
Data formats

All keywords must start in column 1 of the line. The keywords can be up to 8 characters long; but all characters up to column 49 are significant, to include the format qualifier (see below).

The keywords in categories (a) and (b), with the exception of COORDSYS and SPECGRID, have a choice of formats for their associated data:

ARRAY format

Which specifies an array of values that can operate on a designated section of the grid. The values in the array can be assigned directly to the specified grid blocks or nodes, or can modify existing values by addition, subtraction, multiplication or division.

The line of data immediately following the keyword contains information which specifies the type of operation (‘=’, ‘+’, ‘-’, ‘*’, or ‘/’) and the region of the grid over which this operation is to be performed. This line should be terminated with a slash (/).

The remainder of the keyword data consists of the array of values that operate on the designated section of the grid. This array should also be terminated with a slash.

In the data array, asterisks may be used to signify ‘repeat counts’. A data quantity can be repeated a required number of times by preceding it with the required number and an asterisk. There must be no intervening spaces next to the asterisk on either side.

Example:

| PORO | A     | ‘=’ 1 6 1 1 3 3 / 0.15 2*0.14 0.13 0.12 0.11 / |

SINGLE value list format

Which consists of a sequence of lines of data, each specifying a single value that operates on a designated section of the grid. Each line of data contains information that specifies the type of operation, the value of the operand, and the region of the grid over which the operation is to be performed. Each line of data should be terminated with a slash, and the sequence of lines should be terminated with a final line containing just a slash.

Example:

| PORO | S ‘=’ 0.16 / 0.9 2 5 2 3 3 3 / - sets all porosities to 0.16 |

- multiplies selected values by 0.9 |

| ECLIPSE format

Which consists of an array of data to be assigned to the whole grid, terminated with a slash. It is essentially a special case of the Array format, in which the operation is assignment (=) and where it applies to the whole grid. The ECLIPSE format is used for output by FILL (which must of course correspond to the format required for ECLIPSE input), but it can also be used for data input to FILL if the user wishes.
The choice of format is specified by a single letter in up to column 49 of the line containing the keyword, separated from the keyword by one or more blanks: \texttt{A} for ARRAY format, \texttt{S} for Single-value list format, and \texttt{E} for ECLIPSE format. If no letter is present, the ECLIPSE format is assumed by default.

For a complete description of the keyword data formats, the user is referred to the appendix entitled "FILL Data Formats" on page 121, and to the individual keyword descriptions that follow this section in alphabetical order.

Comment lines can be inserted immediately before a keyword. They are identified by two minus signs (\texttt{--}) in columns 1 and 2. Comment lines may not be embedded in the data that follows a keyword, but should be placed between the end of the data and the next keyword. However, it is legal to put additional comments on the same line and following a slash (/) which is used to terminate a data record. Blank lines may be inserted anywhere in the data file.
Missing values

‘Missing values’ can be assigned to selected items of grid data. When the FILL keyword is encountered, the program will attempt to fill in all the missing values by interpolation or other means (e.g. adding dz’s on to depths), as described in the previous section.

Missing values are assigned by n* in the keyword data, where n is an integer denoting the number of consecutive missing values. There must be no blank space between the n and the *.

A single missing value is denoted by 1*, not by a * on its own. For example, an array of grid block porosities, with some missing values, can be entered as

```
PORO A
  *= 1 5 1 3 2 3 /
  0.15 3* 0.20
  5*
  0.20 3* 0.25
  15*0.17 /
```

Missing values can also be assigned by Single-value list format. For example

```
TOPSNODE S
  *= 1* 3 5 1 5 3 3 /
  /
```

assigns a missing value to the top node depths in the specified section of the grid.

For each output variable, if sufficient data has not been provided to eliminate all missing values, then some values will still be ‘missing’ when the FILL program terminates. The missing values will be represented in the output by a constant which is not likely to occur in the user’s data, such as \(-1.0E20\). If any active grid block has a missing value in its data output, the block is made inactive by setting its active cell index to zero. The active cell indices of the grid blocks are output under the keyword ACTNUM (unless this keyword has been suppressed using OUTFILL).
Keywords required

The keyword **SPECGRID** must be present as the first keyword in the FILL data input file. It is required to set up the dimensions and characteristics of the grid. The only exceptions are **DEBUG** and **OUTPUTS**, which can occur anywhere.

The **COORDSYS** keyword is required if there is more than one reservoir, or if the grid completes the circle in radial geometry. If present, it must occur before the first FILL keyword.

The FILL keyword must be used at least once if:

- Any of the keywords **DTHETAV**, **DXV**, **DYV**, **DZ**, **DZCORN** and **DZNODE** are present. These keywords specify the x, y and z (and theta) sizes of the grid blocks, or the distance between nodes in these directions. The FILL process sums these distances to calculate the node coordinates.
- Any of the ‘block centre geometry’ keywords **TOPS**, **Z**, **DEPTH** and **DZ** are present. The FILL process converts this data to corner point form.
- ‘Missing values’ are included in the data. The FILL process ‘fills in’ the missing values by interpolation and other means (e.g. adding dz’s on to depths).

There must be enough data before a FILL keyword to enable the program to construct the (x,y) coordinates, in order to perform the interpolations. The choice of other keywords, and the order in which they appear, will depend on the tasks that the program is intended to perform. The keywords required to perform specific tasks are listed below.

To define x & y positions of the coordinate lines

The location of the coordinate lines is output under the keyword **COORD**. The information necessary to generate this data must be supplied whenever FILL is used to set up a grid in corner point geometry. It is also required if any grid block property arrays need interpolation, because the (x,y) positions of the grid blocks are used in the interpolation procedure.

1. For a regular cartesian grid
   a. **DXV** and **DYV** are normally used. **XV** and **YV** may also be used.
2. For a totally irregular cartesian grid
   a. Use the keyword **COORD**.
3. For a partly irregular cartesian grid
   a. Use a combination of the above keywords.
4. For a radial grid
   a. Use **RADV**, and **THETAV** or **DTHETAV**. Radial grids cannot be irregular.

To define z coordinates of the grid block corners

The z coordinates of the grid block corners are output under the keyword **ZCORN**. The information necessary to generate this data must be supplied whenever FILL is used to set up a grid in corner point geometry.
To interpolate grid block properties

First, sufficient data must be provided to set up the (x,y) coordinate line positions (see above). The required grid block property arrays can then be entered, containing missing values to be filled in by interpolation. Remember that interpolation can only be performed within each layer of blocks, and not from one layer to another. The interpolation is performed when keyword FILL is encountered. The complete arrays of grid block properties are output under their corresponding keywords in Eclipse format. Unwanted output (e.g. the COORD keyword, if grid geometry data is not wanted) can be suppressed with keyword OUTFILL.
To specify and edit grid block properties, without interpolation

Only the appropriate grid block property keywords are required, provided there are no missing values. Individual property keywords may be used more than once, and with different data formats, to perform the editing.
Grid block and node coordinates

All depths and thicknesses are measured along the z axis, which is taken to be vertical, with larger values indicating greater depths. Coordinates on the x axis are taken to increase from left to right, and on the y axis from back to front.

The grid blocks are labelled by 3 numbers i, j and k, where i increases with x, j with y and k with z (for normal geometries). The allowed ranges for these indices are

\[ 1 \leq i \leq \text{NDIVIX} \quad \leq j \leq \text{NDIVIY} \quad \leq k \leq \text{NDIVIZ}. \]

Similarly, the grid nodes are labelled by 3 numbers in, jn and kn, where

\[ 1 \leq \text{in} \leq \text{NDIVIX} + 1 \quad \leq jn \leq \text{NDIVIY} + 1 \quad \leq kn \leq \text{NDIVIZ} + 1 \]

If cylindrical (radial) geometry is used, then r is identified with x and theta with y. R increases towards the right for theta = 0; theta increases in the clockwise direction.
# Keyword summary

A list of all keywords with a brief description of their data is given below. A more complete description of each keyword and its associated data can be found immediately after this section, in alphabetical keyword order.

## Table 4.1 Keyword summary

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Status</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTNUM</td>
<td>Optional</td>
<td>Identifies active grid blocks.</td>
</tr>
<tr>
<td>COORD</td>
<td>Must be set in some way before the FILL keyword</td>
<td>Defines the lines which contain all the grid block corner points for each (i,j) and for each reservoir in the grid.</td>
</tr>
<tr>
<td>COORDSYS</td>
<td>Defaulted when there is only one reservoir</td>
<td>Information about the coordinate system for each reservoir in the grid. Specifies completion of the circle.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Optional</td>
<td>Sets debug output control switches.</td>
</tr>
<tr>
<td>DEPTH</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output.</td>
<td>Depths of grid block centres.</td>
</tr>
<tr>
<td>DTHETAV</td>
<td>Must use THETAV or DTHETAV for cylindrical coordinates.</td>
<td>Vector of theta direction grid block sizes, for each reservoir in the grid.</td>
</tr>
<tr>
<td>DXV</td>
<td>Must use DXV or XV or COORD for cartesian coordinates.</td>
<td>Vector of x direction grid block sizes, for each reservoir in the grid.</td>
</tr>
<tr>
<td>DYV</td>
<td>must use DYV or YV or COORD for cartesian coordinates.</td>
<td>Vector of y direction grid block sizes, for each reservoir in the grid.</td>
</tr>
<tr>
<td>DZ</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output.</td>
<td>Z direction (vertical) grid block sizes.</td>
</tr>
<tr>
<td>DZCORN</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output.</td>
<td>Z direction (vertical) sizes at edges of grid blocks - for faulted dz’s.</td>
</tr>
<tr>
<td>DZNODE</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output.</td>
<td>Z direction (vertical) sizes at the (i,j) grid nodes, for each layer of grid blocks.</td>
</tr>
<tr>
<td>END</td>
<td>Optional</td>
<td>Marks the end of the input to FILL. Not written to the output file.</td>
</tr>
<tr>
<td>FILL</td>
<td>Optional</td>
<td>Causes FILL operation to be performed, does interpolation, and converts block-centred geometry to corner point form</td>
</tr>
<tr>
<td>FLATBLCK</td>
<td>Optional</td>
<td>Sets mode for conversion of block-centred depths to produce flat discontinuous grid blocks.</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>Optional</td>
<td>Insert the contents of a specified file.</td>
</tr>
<tr>
<td>MAPAXES</td>
<td>Optional</td>
<td>Supplies grid origin and axes, to copy to the output file.</td>
</tr>
<tr>
<td>MESSAGES</td>
<td>Optional</td>
<td>Resets message print and stop limits.</td>
</tr>
<tr>
<td>MULTI</td>
<td>Optional</td>
<td>I direction transmissibility multipliers.</td>
</tr>
</tbody>
</table>
### Table 4.1  Keyword summary  (Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Status</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTJ</td>
<td>Optional</td>
<td>j direction transmissibility multipliers.</td>
</tr>
<tr>
<td>MULTK</td>
<td>Optional</td>
<td>K direction transmissibility multipliers.</td>
</tr>
<tr>
<td>MULTR</td>
<td>Optional</td>
<td>R direction transmissibility multipliers.</td>
</tr>
<tr>
<td>MULTTHT</td>
<td>Optional</td>
<td>Theta direction transmissibility multipliers.</td>
</tr>
<tr>
<td>MULTX</td>
<td>Optional</td>
<td>X direction transmissibility multipliers.</td>
</tr>
<tr>
<td>MULTY</td>
<td>Optional</td>
<td>Y direction transmissibility multipliers.</td>
</tr>
<tr>
<td>MULTZ</td>
<td>Optional</td>
<td>Z direction transmissibility multipliers.</td>
</tr>
<tr>
<td>NTG</td>
<td>Optional</td>
<td>Grid block net-to-gross ratios.</td>
</tr>
<tr>
<td>OUTFILL</td>
<td>Optional</td>
<td>Controls which variables are output.</td>
</tr>
<tr>
<td>OUTPUTS</td>
<td>Optional</td>
<td>Controls output switches for all streams except debug and the FILLED stream.</td>
</tr>
<tr>
<td>PERMI</td>
<td>Optional</td>
<td>I direction permeabilities.</td>
</tr>
<tr>
<td>PERMJ</td>
<td>Optional</td>
<td>J direction permeabilities.</td>
</tr>
<tr>
<td>PERMK</td>
<td>Optional</td>
<td>K direction permeabilities.</td>
</tr>
<tr>
<td>PERMR</td>
<td>Optional</td>
<td>R direction permeabilities.</td>
</tr>
<tr>
<td>PERMTHT</td>
<td>Optional</td>
<td>Theta direction permeabilities.</td>
</tr>
<tr>
<td>PERMX</td>
<td>Optional</td>
<td>X direction permeabilities.</td>
</tr>
<tr>
<td>PERMY</td>
<td>Optional</td>
<td>Y direction permeabilities.</td>
</tr>
<tr>
<td>PERMZ</td>
<td>Optional</td>
<td>Z direction permeabilities.</td>
</tr>
<tr>
<td>PORO</td>
<td>Optional</td>
<td>Grid block porosities.</td>
</tr>
<tr>
<td>PORV</td>
<td>Optional</td>
<td>Grid block pore volumes.</td>
</tr>
<tr>
<td>RADV</td>
<td>Must use RADV for cylindrical coordinates.</td>
<td>Vector of radial coordinates of grid nodes, for each reservoir in the grid.</td>
</tr>
<tr>
<td>SLOPBLCK</td>
<td>Optional</td>
<td>Sets mode for conversion of block-centred depths to produce sloping, continuous blocks.</td>
</tr>
<tr>
<td>SPECGRID</td>
<td>Required</td>
<td>Specification of grid dimensions and characteristics.</td>
</tr>
<tr>
<td>THETAV</td>
<td>Must use DTHETAV or THETAV for cylindrical coordinates.</td>
<td>Vector of theta coordinates of grid nodes, for each reservoir in the grid.</td>
</tr>
<tr>
<td>TOPS</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output.</td>
<td>Depths of the centres of the top faces of grid blocks.</td>
</tr>
<tr>
<td>TOPSNODE</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output.</td>
<td>Depths of grid nodes at the top of each layer of grid blocks.</td>
</tr>
<tr>
<td>TRANI</td>
<td>Optional</td>
<td>I direction transmissibilities.</td>
</tr>
<tr>
<td>TRANJ</td>
<td>Optional</td>
<td>J direction transmissibilities.</td>
</tr>
<tr>
<td>TRANK</td>
<td>Optional</td>
<td>K direction transmissibilities.</td>
</tr>
<tr>
<td>TRANR</td>
<td>Optional</td>
<td>R direction transmissibilities.</td>
</tr>
<tr>
<td>TRANTHT</td>
<td>Optional</td>
<td>Theta direction transmissibilities.</td>
</tr>
<tr>
<td>TRANX</td>
<td>Optional</td>
<td>X direction transmissibilities.</td>
</tr>
</tbody>
</table>
**Table 4.1**  
Keyword summary  (Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Status</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANY</td>
<td>Optional</td>
<td>Y direction transmissibilities.</td>
</tr>
<tr>
<td>TRANZ</td>
<td>Optional</td>
<td>Z direction transmissibilities.</td>
</tr>
<tr>
<td>XV</td>
<td>Must use DXV or XV or COORD for cartesian coordinates. Vector of x coordinates of grid nodes, for each reservoir in the grid.</td>
<td></td>
</tr>
<tr>
<td>YV</td>
<td>Must use DYV or YV or COORD for cartesian coordinates. Vector of y coordinates of grid nodes, for each reservoir in the grid.</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output. Depths of grid block centres.</td>
<td></td>
</tr>
<tr>
<td>ZCORN</td>
<td>Sets ZCORN directly. ZCORN must be set in some way for complete geometry output. Depths of grid block corners - for faults.</td>
<td></td>
</tr>
<tr>
<td>ZNODE</td>
<td>Used to set ZCORN. ZCORN must be set in some way for complete geometry output. Depths of grid nodes.</td>
<td></td>
</tr>
</tbody>
</table>
**ACTNUM**  

**Active grid block region**

ACTNUM may be used to make grid blocks inactive. This means that those grid blocks will not be simulated in ECLIPSE.

Possible values are:

0 - inactive
1 - active  

Default value = 1

This is only one of a number of ways of identifying inactive grid blocks. Any method which results in zero pore volume, such as setting the porosity (PORO) or net to gross thickness ratio (NTG) to zero, will cause a grid block to be treated as inactive in ECLIPSE.

In addition, FILL may internally set ACTNUM to zero for a grid block which has bad values for any variable, for instance where a porosity value was left out and not filled in by interpolation.

The formats are described in the section on data formats. The indices are:

- index 1 - `I` - the grid block in the i direction.
- index 2 - `J` - the grid block in the j direction.
- index 3 - `K` - the grid block in the k direction.
Example

Here $NDIVI X = 5$, $NDIVI Y = 3$, $NDIVI Z = 4$.

```
--operator--I1--I2---J1--J2---K1--K2---
ACTNUM A
'=' 1 5 1 3 2 3 /
 0 0 1 0 0
 0 1 1 1 0
 1 1 1 1 1
 15*1 /
--operator--value-----I1--I2---J1--J2---K1--K2---
ACTNUM S
'=' 0 4* 1 1 /
'=' 1 4* 4 4 /
'=' 1 3 3 2 2 1 1 /
/```

**COORD**

**Coordinate lines**

A coordinate line defines the possible positions for corners of grid blocks with the same (i,j) node indices, for grid blocks in a particular reservoir. Given the depth of a particular grid block corner, and the associated coordinate line, the x and y coordinates of the corner can be calculated, as the intersection of the line with a horizontal plane at the specified depth.

A coordinate line is specified by two triplets of x, y and z coordinates, representing two distinct points on it. If the (x,y) coordinates of the top and bottom points are identical, then the z coordinates of the points are not used, and the line is defined to be vertical.

The (x,y) coordinates of the top points in each reservoir are used in the **FILL** operation as a basis for interpolation.

If the z coordinates of the top points on the coordinate lines are not specified, then they default to the depths of the grid nodes of the top layer of the corresponding reservoir (determined by averaging the corner depths). The z coordinates of the bottom points default to the z coordinates of the top points. This defaulting is carried out when the **FILL** keyword is encountered.

**COORD** may only be used with cartesian coordinates, in the input to **FILL**. It may be used with the keywords **DXV**, **DYV**, **XV** and **YV**.

See also the keywords **DTHETAV**, **DXV**, **DYV**, **RADV**, **THETAV**, **XV** and **YV**.

- **UNITS**: metres (**METRIC**), feet (**FIELD**), cm (**LAB**), except for theta and dtheta, which are always in degrees.
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - **ID** - the dimension: 1 - x 2 - y 3 - z
- index 2 - **IP** - the point on a line: 1 - top 2 - bottom
- index 3 - **IN** - the grid node in the i direction.
- index 4 - **JN** - the grid node in the j direction.
- index 5 - **M** - the reservoir.

---

**Example**

Here **NDIVIX = 5**, **NDIVIY = 3**, **NDIVIZ = 4**, **NRES = 2**.

```plaintext
--operator--ID1-ID2--IP1-IP2--IN1-IN2--JN1-JN2--M1--M2---
COORD A
  '=' 1 1 1 1 1 1 6 1 4 2 2 /
  0 10 20 30 40 50
  0 10 20 30 40 50
  0 10 20 30 40 50
  0 10 20 30 40 50 /
--operator--value-----ID1-ID2--IP1-IP2--IN1-IN2--JN1-JN2--M1--M2---
COORD S
  '*' 10.0 1 1 1 1 1 6 1 4 2 2 /
/```

---
COORDSYS  Coordinate system information

Information about the coordinate system for each reservoir in the grid.

The keyword line is followed by NRES data records, each ended with a slash (/). NRES is the number of reservoirs, as specified in SPECGRID.

Each record contains the following data items:
1  Lower bound for block index in k direction (integer)
2  Upper bound for block index in k direction (integer)
3  Completion of circle (in single quotes)
   ‘COMP’ - circle is completed in the theta (or y) direction.
   ‘INCOMP’ - circle is not completed.

For the circle to be completed, the coordinate lines to be identified must be coincident. In the case of cylindrical coordinates, this means that the theta coordinates for JN = 1 and JN = (NDIVIY + 1) must differ by 360 degrees.

• DEFAULT: ‘INCOMP’

If NRES is 1, and completion of the circle is not required, this keyword may be omitted. The bounds of the reservoir will default to the whole grid. See also keywords SPECGRID, COORD, DTHETAV, DXV, DYV, RADV, THETAV, XV, YV.

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
------K1--K2------completed------
COORDSYS
  1  2    ’COMP’  /
  3  4    ’INCOMP’  /
```


**Output control for the DEBUG stream**

This keyword is used to control the output to the debug stream. The keyword line is followed by a list of records, each of which contains the name of a switch and its new value. Each record must be ended by a slash. The list is ended by a record containing just a slash (/). Each record consists of two items, in free format:

1. Switch name (enclosed in single quotes) The names of the switches are given below, with their values.
2. Switch value (an integer)

If a switch is omitted then it retains its previous value.

---

**Switches & values**

**Logging**

Logging, controls the level of logging of the progress of the FILL program. The larger the value, the more output is produced. Each level includes all of the output for the smaller levels.

- **DEFAULT**: 0.

0. No messages, warnings or errors etc. at all.
1. All warnings and errors, compulsory messages and messages for overall phases.
2. Messages for variables.

**Storage**

Debug from storage allocation if > 0.

- **DEFAULT**: 0.

---

**Example**

```
DEBUG
'LOGGING' 4 /
'STORAGE' 1 /
/
```
 Depths of grid block centres

See "z" on page 117.
**DTHETAV**  

**Angular sizes of grid blocks (vector)**

Using this keyword, NDIVIY non-negative real numbers may be supplied for each reservoir. The \( \text{j}^{\text{th}} \) number specifies the angular size in the theta direction of all the grid blocks in that reservoir with that theta axis index - for all \( i \) and \( k \). This means that theta does not vary in the \( i \) or \( k \) directions, within a reservoir.

The only output of this data from the FILL program is provided by COORD. The data must be converted to this form using the FILL keyword.

This keyword can only be used with cylindrical coordinates. It may be used with THETAV, RADV.

See also the keywords SPECGRID, COORD, DXV, DYV, RADV, THETAV, XV, YV, FILL.

- **UNITS**: degrees (METRIC, FIELD or LAB).
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- **index 1 -** \( J \) - the grid block in the \( j \) direction.
- **index 2 -** \( M \) - the reservoir.

**Example**

Here \( \text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, \text{NRES} = 2 \).

```
--operator--J1--J2---M1--M2---
DTHETAV A
  '='  1  3  1  1 /  
  90  90  180 / 
--operator--value-----J1--J2---M1--M2---
DTHETAV S
  '='  120.0  2*  2  2 / 
/
```


X-direction grid block sizes (vector)

Using this keyword, NDIVIX real numbers may be supplied for each reservoir. The ith number specifies the size in the x direction of all the grid blocks in that reservoir with that x axis index - for all j and k. Using this keyword alone, x will not vary in the j or k directions, within a reservoir.

To obtain output from the FILL program for this data, you must use the FILL keyword. Data will be output in the form of COORD.

This keyword may only be used with cartesian coordinates. It may be used with DYV, XV, YV, COORD.

See also the keywords SPECGRID, DTHETAV, DYV, RADV, THETAV, XV, YV, COORD, FILL.

- UNITS: metres (METRIC), feet (FIELD), cm (LAB).
- DEFAULT: <missing>

The formats are described in the section on data formats. The indices are:

index 1 - I - the grid block in the i direction.
index 2 - M - the reservoir.

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--I1--I2---M1--M2---
DXV A
'=' 1 5 1 2 /
5*2000 100 10 10 10 100 /
--operator--value-----I1--I2---M1--M2---
DXV S
'*' 100.0 2* 2 2 /
/
```
DYV

Y direction grid block sizes (vector)

Using this keyword, \textit{NDIVIY} real numbers may be supplied for each reservoir. The $j^{th}$ number specifies the size in the $y$ direction of all the grid blocks in that reservoir with that $y$ axis index - for all $i$ and $k$. Using only this keyword, $y$ will not vary in the $i$ or $k$ directions, within a reservoir.

To obtain output from the FILL program for this data, you must use the \textit{FILL} keyword. The data will be output in the form of \textit{COORD}.

This keyword may only be used with cartesian coordinates. It may be used with \textit{COORD, DXV, XV, YV}.

See also the keywords \textit{SPECGRID, COORD, DTHETAV, DXV, RADV, THETAV, XV, YV, FILL}.

- \textbf{UNITS:} metres (\textit{METRIC}), feet (\textit{FIELD}), cm (\textit{LAB}).
- \textbf{DEFAULT:} <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - $J$ - the grid block in the $j$ direction.
- index 2 - $M$ - the reservoir.

\section*{Example}


\begin{verbatim}
--operator--J1--J2---M1--M2---
DYV A 
  '='  1 3 1 1 / 100 10 100 /
--operator--value-----J1--J2---M1--M2---
DYV S 
  '*' 100.0 2* 1 1 / /
\end{verbatim}
Z direction grid block sizes

The size of each grid block in the z direction may be specified, from the top centre to the bottom centre of the block. The sizes should be non-negative real numbers. The only output from the FILL program for this data is provided by ZCORN. DZ should be converted to this form using the FILL keyword with either the FLATBLCK or the SLOPBLCK conversion options. Sufficient depth data must be supplied, using TOPS, TOPSNODE, Z, ZCORN and/or ZNODE. See also the keywords DZCORN, DZNODE, TOPS, TOPSNODE, Z, ZCORN, ZNODE, FILL, FLATBLCK, SLOPBLCK.

- UNITS: metres (METRIC), feet (FIELD), cm (LAB).
- DEFAULT: <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - I - the grid block in the i direction.
- index 2 - J - the grid block in the j direction.
- index 3 - K - the grid block in the k direction.

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2

```plaintext
--operator--I1--I2---J1--J2---K1--K2---
DZ A
'=' 1 5 1 3 2 3 /
 8 10 15 23 20
10 12 18 25 22
 0 2 5 10 5 /
15*12 /
--operator--value------I1--I2---J1--J2---K1--K2---
DZ S
'=' 20.0 4* 1 1 /
'=' 5.0 4* 4 4 /
'=' 25.0 3 3 2 2 1 1 /
/
```
**DZCORN**

**Z direction grid block sizes at block edges**

Each grid block has 4 edges parallel to the k direction. This keyword enables the z direction sizes at each edge of each grid block to be separately specified. It is used for specifying faults. The quantities should be non-negative real numbers.

To obtain output of this data from the FILL program, sufficient depth data must be supplied (using the keywords TOPS, TOPSNOGE, Z, ZCORN and/or ZNODE); and the FILL keyword must be used. Data will be output in the form of ZCORN.

See also the keywords DZ, DZNODE, TOPS, TOPSNODE, Z, ZCORN, ZNODE, FILL.

- **UNITS:** metres (METRIC), feet (FIELD), cm (LAB).
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

<table>
<thead>
<tr>
<th>Index 1</th>
<th>IC</th>
<th>- the corners in the i direction of a grid block:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>left</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>right</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index 2</th>
<th>I</th>
<th>- the grid block in the i direction.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Index 3</th>
<th>JC</th>
<th>- the corners in the j direction of a grid block:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>back</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>front</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index 4</th>
<th>J</th>
<th>- the grid block in the j direction.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Index 5</th>
<th>K</th>
<th>- the grid block in the k direction.</th>
</tr>
</thead>
</table>

**Example**

Here **NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2**

```
--operator--IC1-IC2--I1--I2---JC1-JC2--J1--J2---K1--K2---
DZCORN A
"=" 1 2 1 2 1 1 3 3 2 2 /
10 15 30 35 /
--operator--value-----IC1-IC2--I1--I2---JC1-JC2--J1--J2---K1--K2---
DZCORN S
"=" 20.0 4* 1 1 2 2 3 3 /
/```

---
**DZNODE**

**Z direction grid block sizes at nodes**

The z direction size of a grid block, at nodes in the i and j directions, and for blocks in the k direction, may be specified using this keyword. The 4 edges around each node, in the 4 adjacent blocks, are given the same z direction size. For the whole of the grid there are \( (\text{NDIVIX}+1)(\text{NDIVIY}+1)\text{NDIVIZ} \) values which may be specified, by non-negative real numbers.

Thicknesses specified using this keyword will be continuous from block to block.

To obtain output of this data from the FILL program, sufficient depth data must be supplied (TOPS, TOPSNODE, Z, ZCORN and/or ZNODE) and the FILL keyword must be used. Data will be output in the form of ZCORN.

See also the keywords DZ, DZCORN, TOPS, TOPSNODE, Z, ZCORN, ZNODE, FILL.

- **UNITS:** metres (METRIC), feet (FIELD), cm (LAB).
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

<table>
<thead>
<tr>
<th>Index</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IN</td>
<td>the grid node in the i direction.</td>
</tr>
<tr>
<td>2</td>
<td>JN</td>
<td>the grid node in the j direction.</td>
</tr>
<tr>
<td>3</td>
<td>K</td>
<td>the grid block in the k direction.</td>
</tr>
</tbody>
</table>

---

**Example**

Here \( \text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, NRES = 2 \).

```
--operator--IN1-IN2--JN1-JN2--K1--K2---
DZNODE A
'='  1   6   1   4   2   3 /
 0 5 10 15 10 5
 0 5 10 15 10 5
 0 5 10 15 10 5
 0 5 10 15 10 5
24*21.0 /
--operator--value-----IN1-IN2--JN1-JN2--K1--K2---
DZNODE S
'='  5.0  4*  1  1 /
'='  25.0  4*  4  4 /
'='  30.0  3  3  2  2  4  4 /
/```

---
End of input data

This keyword signals the end of the input data. No data is entered with this keyword.
Fill in missing values

This keyword causes the FILL operation to be performed. The FILL operation carries out the following tasks:

1. The x and y coordinates of the coordinate lines are determined from the grid geometry data supplied.

2. Missing values in the grid block property keyword data are filled in by interpolation, where possible.

3. The z coordinates of the grid block corners are determined from the geometry data. The z coordinates of the tops and bottoms of the coordinate lines are defaulted to suitable values, wherever this data is missing.

The FILL operation is described in detail in the section entitled "A detailed description" on page 38.

There is no data associated with this keyword.
**FLATBLCK**  

**Flat block mode for geometry conversion**

This keyword sets the flat block mode for the conversion of block-centred depths and thicknesses (TOPS, Z, DZ) to depths and thicknesses at grid block corners. The resulting grid blocks will have flat tops and bottoms, and depths will be discontinuous from one block to the next.

The conversion will take place when the FILL keyword is encountered, so it should be placed before the appropriate FILL keyword.

If neither FLATBLCK nor SLOPBLCK have been specified, the FLATBLCK mode will be assumed as the default conversion mode.

There is no data associated with this keyword.

**Note**  
If this mode is used for geometry conversion, the OLDTRAN keyword should be used in ECLIPSE. If instead NEWTRAN is used in ECLIPSE, a lot of spurious faults will be generated because the depths are not continuous across block boundaries.
**INCLUDE**  

**Name of data file to be included**

The name of the data file to be included at the current position (in quotes unless you are using the EDIT program) should be inserted on the line after the keyword. The data should be terminated by a slash (/).

---

**Example**

```
INCLUDE
'PORO.SECT' /
```

causes FILL to continue input from the file “PORO.SECT”. At the end of that file FILL switches back to the next keyword in the current file.
**MAPAXES**  

**Grid axes with reference to map coordinates**

This keyword defines the grid origin and axes relative to the map coordinates.

If the input data file is prepared by the GRID program, it will contain this keyword. FILL does not process the MAPAXES data, but copies the keyword and data to the FILLED output file, for use if the file is loaded back into the GRID program. ECLIPSE will also read the MAPAXES keyword, and will then include the data in the Grid file that it generates.

The keyword line is followed by one data record, containing 6 real data items and ending with a slash (/).

The data items are:

1. X-coordinate of end of Y-axis
2. Y-coordinate of end of Y-axis
3. X-coordinate of origin
4. Y-coordinate of origin
5. X-coordinate of end of X-axis
6. Y-coordinate of end of X-axis

The MAPAXES keyword may be placed anywhere in the FILL input file. However, it is normally written out by GRID as the first keyword in the file and can be read into FILL before the SPECGRID keyword.

---

**Example**

```
MAPAXES
-- Grid axes with respect to map coordinates
  586494.8  5951371.
  586125.8  5952401.
  587156.0  5952770. /
```
MESSAGES  Resets message print and stop limits

This keyword can be used to reset the print and stop limits for messages of any severity type. Printing of a particular type of message will cease after its print limit has been reached. The run will stop if a particular type of message is generated more times than its stop limit.

There are 6 levels of severity:

1 = message (not an error, purely informative)
2 = comment (probably not a data error)
3 = warning (possibly a data error)
4 = problem (calculation difficulties)
5 = error (definitely a data error)
6 = bug (suspected programming error).

The keyword should be followed by up to 12 integers, terminated by a slash (/). Repeat counts (e.g. 3*1000) and defaults (e.g. 2*) can be used if required. Any items defaulted or left unspecified will not be altered. The items are initialized with their default values.

1  Print limit for severity 1 messages (default = 5000)
2  Print limit for severity 2 messages (default = 1000)
3  Print limit for severity 3 messages (default = 1000)
4  Print limit for severity 4 messages (default = 1000)
5  Print limit for severity 5 messages (default = 1000)
6  Print limit for severity 6 messages (default = 1000)
7  Stop limit for severity 1 messages (default = 5000)
8  Stop limit for severity 2 messages (default = 1000)
9  Stop limit for severity 3 messages (default = 1000)
10 Stop limit for severity 4 messages (default = 100)
11 Stop limit for severity 5 messages (default = 25)
12 Stop limit for severity 6 messages (default = 1)

It is not advisable to alter the stop limits for messages of severity 5 and 6. The MESSAGES keyword can appear anywhere after the SPECGRID keyword.

Example

```
MESSAGES
2* 10 5* 10000 /
```

alters print and stop limits for warnings
I direction transmissibility multipliers

Each transmissibility between a grid block and the next block in the positive i direction may be multiplied by a factor. These factors are specified using this keyword. Thus, a value specified for block (i,j,k) multiplies the transmissibility between blocks (i,j,k) and (i+1,j,k). The values must be real and non-negative.

- **UNITS**: none
- **DEFAULT**: <missing>

The formats are described in the section on data formats.

The indices are:

<table>
<thead>
<tr>
<th>index 1</th>
<th>index 2</th>
<th>index 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>J</td>
<td>K</td>
</tr>
</tbody>
</table>

- the grid block in the i direction.
- the grid block in the j direction.
- the grid block in the k direction.

---

**Example**

Here \( \text{NDIVIX}=5, \text{NDIVIY}=3, \text{NDIVIZ}=4, \text{NRES}=2 \).\n
```
---operator--I1--I2---J1--J2---K1--K2---
MULTI A
'=' 1 5 1 3 2 3 /
0 0.5 1.0 1.0 0
0 0.5 1.0 1.0 0
5*0 15*1.0 /
---operator--value-----I1--I2---J1--J2---K1--K2---
MULTI S
'=' 0.9 3 3 2 2 1 1 /
'*' 0.85 3 3 2 2 1 1 /
/```

---
**MULTJ**

**J direction transmissibility multipliers**

Each transmissibility between a grid block and the next block in the positive j direction may be multiplied by a factor. These factors are specified using this keyword. Thus, a value specified for block (i,j,k) multiplies the transmissibility between blocks (i,j,k) and (i,j+1,k). The values must be real and non-negative.

- **UNITS:** none
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I - the grid block in the i direction.</td>
</tr>
<tr>
<td>2</td>
<td>J - the grid block in the j direction.</td>
</tr>
<tr>
<td>3</td>
<td>K - the grid block in the k direction.</td>
</tr>
</tbody>
</table>

**Example**

Here \(\text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, \text{NRES} = 2.\)

```plaintext
--operator--I1--I2---J1--J2---K1--K2---
MULTJ A
'=' 1 5 1 3 2 3 /
 0 0.5 1.0 1.0 0
 0 0.5 1.0 1.0 0
 5*0 15*1.0 /
--operator--value----I1--I2---J1--J2---K1--K2---
MULTJ S
'=' 0.9 3 3 2 2 1 1 /
'*' 0.85 3 3 2 2 1 1 /
/
```
**MULTK**  

**K direction transmissibility multipliers**

Each transmissibility between a grid block and the next block in the positive k direction may be multiplied by a factor. These factors are specified using this keyword. Thus, a value specified for block (i,j,k) multiplies the transmissibility between blocks (i,j,k) and (i,j,k+1). The values must be real and non-negative.

- **UNITs**: none
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- `MULTK A`  
  
  --operator--I1--I2---J1--J2---K1--K2---
  MULTK A
  'A' 1 5 1 3 2 3 /  
  0 0.5 1.0 1.0 0  
  0 0.5 1.0 1.0 0  
  5*0 15*1.0 /  
  --operator--value-----I1--I2---J1--J2---K1--K2---
  MULTK S
  'S' 0.9 3 3 2 2 1 1 /  
  '*' 0.85 3 3 2 2 1 1 /  
  /  

**Example**

Here `NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2`. 

---operator--I1--I2---J1--J2---K1--K2---
MULTK A
'=' 1 5 1 3 2 3 /  
0 0.5 1.0 1.0 0  
0 0.5 1.0 1.0 0  
5*0 15*1.0 /  
---operator--value-----I1--I2---J1--J2---K1--K2---
MULTK S
'=' 0.9 3 3 2 2 1 1 /  
'*' 0.85 3 3 2 2 1 1 /  
/
R direction transmissibility multipliers

See "MULTI" on page 79.
MULTTHT  theta direction transmissibility multipliers

See "MULTJ" on page 80.
MULTX  X direction transmissibility multipliers

See "MULTI" on page 79.
MULTY  Y direction transmissibility multipliers

See "MULTJ" on page 80.
MULTZ  Z direction transmissibility multipliers

See "MULTK" on page 81.
**NTG**  

**Net to gross ratios**

The values specified with this keyword are used in ECLIPSE to convert from gross to net thicknesses. They act as multipliers for grid block pore volumes and transmissibilities in the i and j directions. They must be non-negative.

- **UNITS** none
- **DEFAULT:** \(<\text{missing}>\)

The formats are described in the section on data formats. The indices are:

- index 1 - I - the grid block in the i direction.
- index 2 - J - the grid block in the j direction.
- index 3 - K - the grid block in the k direction.

**Example**

Here \(\text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, \text{NRES} = 2\).

```
--operator--I1--I2---J1--J2---K1--K2---
NTG A
  '=' 1 5 1 3 2 3 /
  0 0.5 1.0 1.0 0
  0 0.5 1.0 1.0 0
  5*0 15*1.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
NTG S
  '=' 0.9 3 3 2 2 1 1 /
  '*' 0.85 3 3 2 2 1 1 /
```

/
OUTFILL  Output control for the FILLED stream

This keyword can be used to select the variables that are to be written to the FILLED output file.

If this keyword is omitted, the output will contain the information associated with each input keyword. The grid geometry data is output under the keywords COORD (which specifies the x, y and z coordinates of the tops and bottoms of the coordinate lines) and ZCORN (which specifies the depths of the grid block corners). Grid block property data is output under the appropriate property keyword. In addition, the active cell indices are output by default under the keyword ACTNUM. If sufficient data has not been provided to define the output quantities fully, then the grid blocks with undetermined values will have their active cell indices set to zero. Any output quantities that remain undetermined will appear as -1.0E20.

No output is produced until all the input data has been read. To turn off the output of a variable, the OUTFILL keyword should be placed at the end of the data, after the final FILL keyword.

The following keywords will be output by default:

- ACTNUM, COORDSYS, SPECGRID.

The following keywords will be output by default if present in the input file, or if an alias for the keyword is present:

- MULTI, MULTJ, MULTK, NTG, PERMI, PERMJ, PERMK, PORO, PORV, TRANI, TRANJ, TRANK.

The keyword COORD will be output by default if (x,y) or (r,theta) coordinate data are present in the input file. The keyword ZCORN will be output by default if depths are present in the input file.

In addition, the following keywords may be selected for output: DZ, DZCORN, DZNODE, TOPS, TOPSNODE, Z.

The keyword line is followed by a list of records, each of which sets the output status for a single keyword. Each record must be ended by a slash. The list is ended by a record containing just a slash (/). Each record consists of two items, in free format:

1  Keyword (enclosed in single quotes)

   Any keyword in FILL input, for which there is associated output, may be used.
   In addition, the keyword ‘ALL’ may be used to set or clear all of the other keyword switches simultaneously.

2  Output status (enclosed in single quotes)

   ‘YES’ - output is required.
   ‘NO’ - output is not required.
   ‘Y’ and ‘N’ are acceptable instead of ‘YES’ and ‘NO’.

Example

```
OUTFILL
   'ALL' 'NO' /
   'ACTNUM' 'YES' /
   'SPECGRID' 'YES' /
   'PORO' 'YES' /
/
```
OUTPUTS

Output control (except DEBUG and FILLED)

This keyword controls output to all streams except for DEBUG and FILLED.
The keyword line is followed by a list of records, each of which contains the name of a switch and its new value. Each record must be ended by a slash. The list is ended by a record containing just a slash (/). Each record consists of two items, in free format:
1 Switch name (enclosed in single quotes)
   The names of the switches are given below, with their values.
2 Switch value (an integer)
   If a switch is omitted then it retains its previous value.

Switches and values

LOGTERM

LOGTERM controls the level of logging of the progress of FILL to the terminal stream, via MESSAGEs.
Note that warnings and errors etc. are compulsory.
The larger the value, the more output is produced. Each level includes all output for the lower levels.

- DEFAULT: 0
  0 Only compulsory messages.
  1 Messages for overall phases.
  2 Messages for variables.
  3 Messages for reservoirs.
  4 Messages for layers.

LOGPRINT

LOGPRINT controls the level of logging to the print stream. See LOGTERM.

GRIDFILE

GRIDFILE switch for generation of grid geometry file for use in graphics

- DEFAULT: 0
  0 - no grid file.
  1 - grid file.
Example

OUTFUTS
'LOGTERM' 1 /
'LOGPRINT' 2 /
'GRIDFILE' 1 /
/
The i direction absolute permeability for each grid block may be specified using this keyword. The values must be non-negative real numbers.

- **UNITS**: millidarcies (METRIC, FIELD or LAB)
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- **index 1** - I - the grid block in the i direction.
- **index 2** - J - the grid block in the j direction.
- **index 3** - K - the grid block in the k direction.

### Example

Here `NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2`.

```
--operator--I1--I2---J1--J2---K1--K2---
PERMI A
'='  1 5 1 3 2 3 /
10 20 25 20 15
15 30 35 30 10
5 25 30 20 5
15*50.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
PERMI S
'='  10.0 3 3 2 2 1 1 /
'*'  0.85 3 3 2 2 1 1 /
/```
PERMJ  J direction permeabilities

The j direction absolute permeability for each grid block may be specified using this keyword. The values must be non-negative real numbers.

- **UNITS:** millidarcies (METRIC, FIELD or LAB)
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - I - the grid block in the i direction.
- index 2 - J - the grid block in the j direction.
- index 3 - K - the grid block in the k direction.

**Example**

Here \( \text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, \text{NRES} = 2. \)

```plaintext
--operator--I1--I2---J1--J2---K1--K2---
PERMJ A
'=' 1 5 1 3 2 3 /
10 20 25 20 15
15 30 35 30 10
5 25 30 20 5
15*50.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
PERMJ S
'=' 10.0 3 3 2 2 1 1 /
'*' 0.85 3 3 2 2 1 1 /
/
```
**PERMK**

**K direction permeabilities**

The k direction absolute permeability for each grid block may be specified using this keyword. The values must be non-negative real numbers.

- **UNITS:** millidarcies ([METRIC, FIELD or LAB])
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

- **index 1 -** I - the grid block in the i direction.
- **index 2 -** J - the grid block in the j direction.
- **index 3 -** K - the grid block in the k direction.

### Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--I1--I2---J1--J2---K1--K2---
PERMK A
'=' 1 5 13 2 3 /
10 20 25 20 15
15 30 35 30 10
5 25 30 20 5
15*50.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
PERMK S
'='* 10.0 3 3 2 2 1 1 /
'** 0.85 3 3 2 2 1 1 /
/
```
PERMR  R direction permeabilities

See "PERMI" on page 91.
PERMHT  Theta direction permeabilities

See "PERMJ" on page 92.
**PERMX**  

**X direction permeabilities**

See "PERMI" on page 91.
PERMY  Y direction permeabilities

See "PERMJ" on page 92.
**PERMZ**  

Z direction permeabilities

See "PERMK" on page 93.
Grid block porosities

The grid block porosities are specified using this keyword. They must be non-negative real numbers.

- **UNITS**: none
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- **index 1 - I** - the grid block in the i direction.
- **index 2 - J** - the grid block in the j direction.
- **index 3 - K** - the grid block in the k direction.

**Example**

Here $NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2$.

```
--operator--I1--I2---J1--J2---K1--K2---
PORO A
'=' 1 5 1 3 2 3 /
 0.15 3* 0.20
 5*
 0.20 3* 0.25
 15*0.17 /
--operator--value-----I1--I2---J1--J2---K1--K2---
PORO S
'=' 0.20 3 3 2 2 1 1 /
'**' 0.85 3 3 2 2 1 1 /
/
```
Grid block pore volumes

The grid block pore volumes may be specified using this keyword. They must be non-negative real numbers.

- **UNITS:** reservoir metres$^3$ (METRIC), barrels (FIELD), cm$^3$ (LAB).
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - $I$ - the grid block in the $i$ direction.
- index 2 - $J$ - the grid block in the $j$ direction.
- index 3 - $K$ - the grid block in the $k$ direction.

**Example**

Here $\text{NDIVIX} = 5$, $\text{NDIVIY} = 3$, $\text{NDIVIZ} = 4$, $\text{NRES} = 2$.

```
--operator--I1--I2---J1--J2---K1--K2---
PORV A
'='  1  5  1  3  2  3 /
  0 0.5 1.0 1.0 0
  0 0.5 1.0 1.0 0
  5*0 15*1.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
PORV S
'='  0.9  3  3  2  2  1  1 /
'*'  0.85  3  3  2  2  1  1 /
'*'  1.0E8 /
/```

RADV  R coordinates of grid nodes (vector)

This keyword provides a way of specifying the r coordinates of grid nodes, using just one vector of positive real numbers for each reservoir. The in’th value in the vector is the r coordinate for grid nodes (in,jn,kn) in that reservoir, for all suitable jn and kn.

This keyword may only be used with cylindrical coordinates. It may be used with DTHETAV and THETAV.

See also the keywords SPECGRID, COORD, DTHETAV, DXV, DYV, THETAV, XV, YV.

- UNITS: metres (METRIC), feet (FIELD), cm (LAB).
- DEFAULT: <missing>

The formats are described in the section on data formats. The indices are:

index 1 - IN - the grid node in the i direction.
index 2 - M - the reservoir.

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--IN1-IN2--M1--M2---
RADV A
  '='  1  6  1  2 /
  5 10 20 40 80 160
  5 10 20 40 80 160 /
--operator--value-----IN1-IN2--M1--M2---
RADV S
  '*'  0.9 /
/
```
SLOPBLCK  Sloping block mode for geometry conversion

This keyword sets the sloping block mode for the conversion of block-centred depths and thicknesses (TOPS, Z, DZ) to depths and thicknesses at grid block corners. The resulting grid blocks will have sloping top and bottom surfaces, and depths will be continuous across block boundaries. The conversion will take place when the FILL keyword is encountered, so it should be placed before the appropriate FILL keyword. If neither FLATBLCK nor SLOPBLCK have been specified, the FLATBLCK mode will be assumed as the default conversion mode. There is no data associated with this keyword.

Note  If the SLOPBLCK mode is used for geometry conversion, the NEWTRAN keyword can be used in ECLIPSE. It provides a more accurate calculation of transmissibilities, using cell corner positions, and takes account of faults if these have been specifically introduced by the user.
**SPECGRID**

**Specification of grid characteristics**

This keyword sets the dimensions and other characteristics of the grid as a whole. It must be the first keyword in the input file (apart from a MAPAXES keyword).

Although the SPECGRID keyword is necessary for the operation of FILL, it is not necessary in ECLIPSE as the dimensioning information is supplied with the keyword RUNSPEC. The output of SPECGRID from FILL can be suppressed with the OUTFILL keyword. If SPECGRID is present in the input to ECLIPSE, its associated data must agree with the data in RUNSPEC.

One of the dimensions mentioned below is the number of reservoirs. Each reservoir has its own coordinate system, defined by its own set of coordinate lines and the information supplied by the COORDSYS keyword. Normally only one reservoir should be used - refer to the section on multiple reservoirs for the use of more than one.

See also the keywords COORDSYS, COORD, DTHETAV, DXV, DYV, RADV, THETAV, XV, YV.

Following the keyword line there is a single record containing the following items, and ended with a slash (/):

1. **NDIVIX** The number of grid blocks in the i (x or r) direction.
   - DEFAULT: 1
2. **NDIVIY** The number of grid blocks in the j (y or theta) direction.
   - DEFAULT: 1
3. **NDIVIZ** The number of grid blocks in the k (z or depth) direction.
   - DEFAULT: 1
4. **NRES** The number of reservoirs.
   - DEFAULT: 1
5. **QRDIAL**
   - T - cylindrical (radial) coordinates.
   - F - cartesian coordinates.
   - DEFAULT: F
   - The T or F should **not** be enclosed in quotes.

---

**Example**

```plaintext
SPECGRID
5 3 4 2 T /
```
**THETAV**

**Theta coordinates of grid nodes (vector)**

This keyword provides a way of specifying the theta coordinates of grid nodes, using just one vector of positive real numbers for each reservoir. The jn’th value in the vector is the theta coordinate for grid nodes (in,jn,kn) in that reservoir, for all suitable in and kn.

This keyword may only be used with cylindrical coordinates. It may be used with DTHETAV and RADV.

See also the keywords SPECGRID, COORD, DTHETAV, DXV, DYV, RADV, XV, YV.

- **UNITS:** degrees (METRIC, FIELD, LAB).
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - JN - the grid node in the j direction.
- index 2 - M - the reservoir.

### Example

Here  NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--JN1-JN2--M1--M2---
THETAV A
'=' 1 4 1 2 /
0 120 240 360
0 30 60 90 /
--operator--value-----JN1-JN2--M1--M2---
THETAV S
'*' 1.23 /
/
```

---
**TOPS**

**Depths of top centre of each grid block**

With this keyword the depth of the top face of each grid block may be specified, at the centre of the face.

The only output for this data from the FILL program is in the form of ZCORN, and data must be converted to this form using the FILL keyword with either the FLATBLCK or the SLOPBLCK conversion options.

See also the keywords DZ, DZCORN, DZNODE, TOPSNODE, Z, ZCORN, ZNODE, FILL, FLATBLCK, SLOPBLCK.

- **UNITS**: metres (METRIC), feet (FIELD), cm (LAB).
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

<table>
<thead>
<tr>
<th>index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I - the grid block in the i direction.</td>
</tr>
<tr>
<td>2</td>
<td>J - the grid block in the j direction.</td>
</tr>
<tr>
<td>3</td>
<td>K - the grid block in the k direction.</td>
</tr>
</tbody>
</table>

**Example**

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--I1--I2---J1--J2---K1--K2---
TOPS A
'='  1  5  1  3  2  3 /
  1000 3* 1500
  5*
 1200 3* 1700
  15*2500 /
--operator--value-----I1--I2---J1--J2---K1--K2---
TOPS S
'='  700.0 3  3  2  2  1  1 /
'+'  45.0 3  3  2  2  2  2 /
/```


TOPSNODE

 Depths of block tops at nodes

The depths of the top of each layer of blocks, at each node in the i and j directions, may be specified using this keyword. A grid node in a layer of nodes corresponds to 4 grid block corners, provided it is not at the edge of the reservoir. Specification of all the depths using this keyword results in a non-faulted field with disconnected layers (in general). Output of this data from the FILL program will be in the form of ZCORN. See also the keywords DZ, DZCORN, DZNODE, TOPS, Z, ZCORN, ZNODE.

- UNITS: metres (METRIC), feet (FIELD), cm (LAB).
- DEFAULT: <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - IN - the grid node in the i direction.
- index 2 - JN - the grid node in the j direction.
- index 3 - K - the grid node in the k direction.

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--IN1-IN2--JN1-JN2--K1--K2---
TOPSNODE A
'= ' 1 6 1 4 2 3 /
1000 4* 1200
6* 1300 4* 1500
24*2000 /

--operator--value-----IN1-IN2--JN1-JN2--K1--K2---
TOPSNODE S
'= ' 3000 3 3 2 2 4 4 /
'= ' 23.0 3 3 2 2 2 2 /
/```

---
TRANI

I direction transmissibilities

Each transmissibility between a grid block and the next block in the positive i direction may be specified using this keyword. Thus, a value specified for block (i,j,k) is the transmissibility between blocks (i,j,k) and (i+1,j,k). The values must be real and non-negative.

- **UNITS:** cp.m³/day/bar (METRIC)
  - cp.bbl/day/psi (FIELD)
  - cp.cm³/hour/atm (LAB).
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

<table>
<thead>
<tr>
<th>index 1</th>
<th>I</th>
<th>- the grid block in the i direction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>index 2</td>
<td>J</td>
<td>- the grid block in the j direction.</td>
</tr>
<tr>
<td>index 3</td>
<td>K</td>
<td>- the grid block in the k direction.</td>
</tr>
</tbody>
</table>

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--I1--I2---J1--J2---K1--K2---
TRANI A
'='  1  5  1  3  2  3 /
 0  0.5  1.0  1.0  0
 0  0.5  1.0  1.0  0
 5*0
 15*1.0 /
--operator--value------I1--I2---J1--J2---K1--K2---
TRANI S
'**'  50000 /
/
```
J direction transmissibilities

Each transmissibility between a grid block and the next block in the positive j direction may be specified using this keyword. Thus, a value specified for block \((i,j,k)\) is the transmissibility between blocks \((i,j,k)\) and \((i+1,j,k)\). The values must be real and non-negative.

- **UNITS:** \(\text{cp.m}^3/\text{day/bar (METRIC)}\)
  \(\text{cp.bbl/day/psi (FIELD)}\)
  \(\text{cp.cm}^3/\text{hour/atm (LAB)}\).

- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

- **index 1** - I - the grid block in the i direction.
- **index 2** - J - the grid block in the j direction.
- **index 3** - K - the grid block in the k direction.

**Example**

Here \(\text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, \text{NRES} = 2\).

```
--operator--I1--I2---J1--J2---K1--K2---
TRANJ A
  = 1 5 1 3 2 3 /
  0 0.5 1.0 1.0 0
  0 0.5 1.0 1.0 0
  5*0
  15*1.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
TRANJ S
  *= 50000 /
/
```

--operator--I1--I2---J1--J2---K1--K2---
TRANJ A
  = 1 5 1 3 2 3 /
  0 0.5 1.0 1.0 0
  0 0.5 1.0 1.0 0
  5*0
  15*1.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
TRANJ S
  *= 50000 /
/
**TRANK**

### K direction transmissibilities

Each transmissibility between a grid block and the next block in the positive k direction may be specified using this keyword. Thus, a value specified for block (i,j,k) is the transmissibility between blocks (i,j,k) and (i+1,j,k). The values must be real and non-negative.

- **UNITS:** \( \text{cm}^3/\text{day/bar} \) (METRIC)
  
  \( \text{cp.bbl/day/psi} \) (FIELD)
  
  \( \text{cp.cm}^3/\text{hour/atm} \) (LAB).

- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I</td>
</tr>
<tr>
<td>2</td>
<td>J</td>
</tr>
<tr>
<td>3</td>
<td>K</td>
</tr>
</tbody>
</table>

**Example**

Here \( \text{NDIVIX} = 5, \text{NDIVIY} = 3, \text{NDIVIZ} = 4, \text{NRES} = 2 \).

```
--operator--I1--I2---J1--J2---K1--K2---
TRANK A
'='    1  5  1  3  2  3 /
  0 0.5 1.0 1.0 0
  0 0.5 1.0 1.0 0
  5*0
 15*1.0 /
--operator--value-----I1--I2---J1--J2---K1--K2---
TRANK S
'**'    50000 /
/
```

R direction transmissibilities

See "TRANR" on page 107.
TRANTHT  Theta direction transmissibilities

See "TRANJ" on page 108.
TRANX  X direction transmissibilities

See "TRANI" on page 107.
TRANY

Y direction transmissibilities

See "TRANJ" on page 108.
Z direction transmissibilities

See "TRANK" on page 109.
**XV**

**X coordinates of grid nodes (vector)**

This keyword provides a way of specifying the x coordinates of grid nodes, using just one vector of real numbers for each reservoir. The $i^{th}$ value in the vector is the x coordinate for grid nodes $(i,n,k)$ in that reservoir, for all suitable $j$ and $k$.

This keyword may only be used with cartesian coordinates. It may be used with DXV, DYV, YV, COORD.

See also the keywords SPECGRID, COORD, DTHETA, DXV, DYV, RADV, THETA, YV.

- **UNITS**: metres (METRIC), feet (FIELD), cm (LAB).
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- **index 1** - \textit{IN} - the grid node in the $i$ direction.
- **index 2** - \textit{M} - the reservoir.

### Example

Here NDIVI = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--IN1-IN2--M1--M2---
XV A
'=' 1 6 1 2 /
1000 2000 4000 5000 5500 6000
0 100 200 300 400 500 /
--operator--value-----IN1-IN2--M1--M2---
XV S
'+' 55.0 1 1 /
/
```


**YV**

**Y coordinates of grid nodes (vector)**

This keyword provides a way of specifying the y coordinates of grid nodes, using just one vector of real numbers for each reservoir. The $j^n$th value in the vector is the y coordinate for grid nodes $(i_n,j_n,k_n)$ in that reservoir, for all suitable $i_n$ and $k_n$.

This keyword may only be used with cartesian coordinates. It may be used with $DXV$, $DYV$, $XV$, $COORD$.

See also the keywords $SPECGRID$, $COORD$, $DTHETA$, $DXV$, $DYV$, $RADV$, $THETA$, $XV$.

- **UNITS**: metres (METRIC), feet (FIELD), cm (LAB).
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- **index 1**: $JN$ - the grid node in the $j$ direction.
- **index 2**: $M$ - the reservoir.

---

**Example**

Here $NDIVIX = 5$, $NDIVIY = 3$, $NDIVIZ = 4$, $NRES = 2$.

```
--operator--JN1-JN2--M1--M2---
YV A
  '=' 1  4  1  2 /
  1000 2000 4000 4500
  0 100 200 400 /
--operator--value-----JN1-JN2--M1--M2---
YV S
  '+' 55.0 1 1 /
/```

**Depths of grid block centres**

With this keyword the depth of the centre of each grid block may be specified. The values must be real. The only output for this data from the FILL program is in the form of ZCORN, and data must be converted to this form using the FILL keyword with either the FLATBLCK or the SLOPBLCK conversion options.

See also the keywords DZ, DZCORN, DZNODE, TOPS, TOPSNODE, ZCORN, ZNODE, FILL, FLATBLCK, SLOPBLCK.

- **UNITS:** metres (METRIC), feet (FIELD), cm (LAB).
- **DEFAULT:** <missing>

The formats are described in the section on data formats. The indices are:

- **index 1:** I - the grid block in the i direction.
- **index 2:** J - the grid block in the j direction.
- **index 3:** K - the grid block in the k direction.

**Example**

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--I1--I2---J1--J2---K1--K2---
Z A
'=+' 1 5 1 3 2 3 /
1000 3* 1500
5*
1200 3* 1700
15*2500 /
--operator--value-----I1--I2---J1--J2---K1--K2---
Z S
'=+' 700.0 3 3 2 2 1 1 /
'+=' 45.0 3 3 2 2 2 2 /
/```

---
**ZCORN**

**Depths of grid block corners**

Each grid block has 8 corners. This keyword enables the depths of each corner of each grid block to be separately specified. It is used for specifying faults, and as the output of depths from the FILL program to ECLIPSE.

See also the keywords **DZ, DZCORN, DZNODE, TOPS, TOPSNODE, Z, ZNODE, FILL**.

- **UNITS**: metres (**METRIC**), feet (**FIELD**), cm (**LAB**).
- **DEFAULT**: <missing>

The formats are described in the section on data formats. The indices are:

- **index 1 - IC**: the corners in the i direction of a grid block:
  - 1 - left
  - 2 - right
- **index 2 - I**: the grid block in the i direction.
- **index 3 - JC**: the corners in the j direction of a grid block:
  - 1 - back
  - 2 - front
- **index 4 - J**: the grid block in the j direction.
- **index 5 - K**: the grid block in the k direction.

**Example**

Here **NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2**.

```
---operator--IC1-IC2--I1--I2---JC1-JC2--J1--J2---KC1-KC2--K1--K2---
ZCORN A
'+'  1  2  1  2  1  1  3  3  1  2  2  2 /
  10 15 30 35 /
-ZCORN S
'-' 20.0 2* 1  4  2*  2  3  2*  3  3 /
```

---operator--IC1-IC2--I1--I2---JC1-JC2--J1--J2---KC1-KC2--K1--K2---
ZNODE

Depths of grid nodes

The depth of each grid node in the grid may be specified using this keyword. A grid node corresponds to 8 grid block corners, provided it is not on the edge of the reservoir. So specifying all depths using this keyword results in a non-faulted field with contiguous grid blocks.

ZNODE is useful primarily for setting up simple benchmark problems, rather than for large field studies.

Output of this data from the FILL program will be in the form of ZCORN.

See also the keywords DZ, DZCORN, DZNODE, TOPS, TOPSNODE, Z, ZCORN.

• UNITS: metres (METRIC), feet (FIELD), cm (LAB).
• DEFAULT: <missing>

The formats are described in the section on data formats. The indices are:

- index 1 - IN - the grid node in the i direction.
- index 2 - JN - the grid node in the j direction.
- index 3 - K - the grid node in the k direction.

Example

Here NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 2.

```
--operator--IN1-IN2--JN1-JN2--KN1-KN2--
ZNODE A
'=' 1  6 1 4 2 3 /
 1000 4* 1200
 12*
 1300 4* 1500
 24*2000 /
--operator--value-----IN1-IN2--JN1-JN2--KN1-KN2--
ZNODE S
'=' 4000 3 3 2 2 5 5 /
'-' 23.0 3 3 2 2 2 2 /
/```


Introduction

There are three data formats for the grid geometry keywords and the grid block property keywords in the input to FILL.

The choice of format is specified by a single letter in up to column 49 of the line containing the keyword, and separated from it by one or more blanks.

A - ARRAY format.
S - SINGLE value list format.
E - ECLIPSE input format.

If there is no character in that position, the ECLIPSE format is assumed by default.

For any data item in the three formats, repeat counts may be used for repeated values (e.g. 12*13.7). Spaces must not be inserted on either side of the asterisk. <Missing> values may be entered using a repeat count followed by a blank (e.g. 5* or 1*). All data must be specified in some way, either as normal data or as missing values, otherwise an error message is given.

The three data formats are described below.
The line following the keyword line contains several items describing the amount of `ARRAY` type data to be input, and whether the data is to replace or modify the original data. This line of descriptive data is terminated by a slash (/). There can be up to 13 items on the line, arranged in free format:

1. Operator (enclosed in single quotes)
   - ‘=’ - assign.
   - ‘+’ - add.
   - ‘*’ - multiply.
   - ‘-’ - subtract.
   - ‘/’ - divide.

   The old value is replaced by the new value in the case of assignment, or by \(<\text{old value}> <\text{operator}> <\text{new value}>\) for the other operators. For integer grid variables (`ACTNUM`), only the assignment operator is allowed.

2. Lower bound for index 1 (integer)
3. Upper bound for index 1 (integer)
4. Lower bound for index 2 (integer)
5. Upper bound for index 2 (integer)
6. Lower bound for index 3 (integer)
7. Upper bound for index 3 (integer)
8. Lower bound for index 4 (integer)
9. Upper bound for index 4 (integer)
10. Lower bound for index 5 (integer)
11. Upper bound for index 5 (integer)
12. Lower bounds for index 6 (integer)
13. Upper bounds for index 6 (integer)

The index bounds specify the blocks, nodes or corners that will be modified by the data associated with the keyword. There can be up to 6 separate indices, but many keywords have only 3. The set of indices associated with each keyword is described elsewhere in this manual, in the keyword descriptions. As an illustrative example, the keyword `PORO` has 3 indices:

   - index 1 = I location of grid block
   - index 2 = J location of grid block
   - index 3 = K location of grid block

   The bounds must always satisfy

   \[ \min \text{ lower bound} \leq \text{lower bound} \leq \text{upper bound} \leq \max \text{ upper bound} \]

   In the case of porosity this means

   \[ 1 \leq I_1 \leq I_2 \leq \text{NDIVIX} \]
   \[ 1 \leq J_1 \leq J_2 \leq \text{NDIVIY} \]
   \[ 1 \leq K_1 \leq K_2 \leq \text{NDIVIZ} \]
Items on the descriptive data line can be defaulted, either by ending the line prematurely with a slash (in which case all data remaining unspecified will be defaulted), or by defaulting individual items in the same way as <missing> values are declared (1*, 2* etc.).

The operator must be specified - it cannot be defaulted.

A lower index bound defaults to the lowest value possible for that index, generally 1.

An upper index bound defaults to the value of the lower bound, if the lower bound has been specified. If the lower bound has also been defaulted, the upper bound defaults to its highest possible value (e.g. NDIVIX for the grid block index i).

The descriptive data line is followed by a stream of real or integer values in free format, terminated by a slash (/). Integer values are expected for ACTNUM; real values are expected for the other grid keywords. Repeat counts may be used for repeated values (e.g. 12*13.7).<Missing> values may be entered using a repeat count followed by a blank (e.g. 5* or 1*).

Each value is associated with a grid block, node or corner (depending on the particular keyword) with indices lying between the bounds specified on the descriptive data line. The sequence of values entered is ordered so that the first index varies most rapidly, and the last index least rapidly. The lower and upper bounds for each index determine the number of values expected.

For instance, the number of data values required for PORO is in general

\[(I_2-I_1+1) \times (J_2-J_1+1) \times (K_2-K_1+1)\]

where “1” refers to lower bound and “2” to upper bound. If an insufficient number of values are supplied, the values remaining unspecified will be defaulted to <missing>.

Example

Assume NDIVIX = 5, NDIVIY = 3, NDIVIZ = 4, NRES = 1 in these examples.

```
--operator--I1--I2---J1--J2---K1--K2---
PORO A
  '='          2  5  1  3  4  4 /
.100000 .150000 .140000 .130000
.180000 .170000 .160000 .145000
.130000 .120000 .110000 .100000
/
```


ECLIPSE format

The ECLIPSE format is the form in which the FILL program writes variables to output, for input to ECLIPSE. It is essentially a special case of the ARRAY format, in which the operator is ‘=’ and the indices cover their maximum range. In the ECLIPSE format the descriptive data line is omitted. The keyword line is followed by a sequence of values (including repeat counts and <missing> values) sufficient to cover the entire range of indices. The data is terminated with a slash (/).

Example

| PORO | .100000 .100000 .150000 .140000 .130000 |
|      | .100000 .180000 .170000 .160000 .145000 |
|      | .100000 .130000 .120000 .110000 .100000 |
|      | .100000 .100000 .150000 .140000 .130000 |
|      | .100000 .180000 .170000 .160000 .145000 |
|      | .100000 .130000 .120000 .110000 .100000 |
|      | .100000 .100000 .150000 .140000 .130000 |
|      | .100000 .180000 .170000 .160000 .145000 |
|      | .100000 .130000 .120000 .110000 .100000 |
|      | .100000 .100000 .150000 .140000 .130000 |
|      | .100000 .180000 .170000 .160000 .145000 |
|      | .100000 .130000 .120000 .110000 .100000 |

/
SINGLE value list format

The Single-value list format allows a single data value to operate on all the grid blocks, nodes or corners (depending on the particular keyword) with indices lying within a specified range.

The keyword line is followed by a list of data records, each terminated by a slash (/). The list is ended by a final record containing just a slash. The items in each data record specify the type of operation (assign, add etc.), the single data value, and the range of indices over which the operation is to be performed. The items are arranged in free format as follows:

1. Operator (enclosed in single quotes)
   - ‘=’ - assign.
   - ‘+’ - add.
   - ‘*’ - multiply.
   - ‘-’ - subtract.
   - ‘/’ - divide.

   The old value is replaced by the new value in the case of assignment, or by \(( \text{<old value>} <\text{operator}> <\text{new value}> )\) for the other operators. For integer grid variables (ACTNUM), only the assignment operator is allowed.

2. Data value (real or integer according to the keyword)
   A \(<\text{missing}>\) value (1*) can be assigned.

3. Lower bound for index 1 (integer)
4. Upper bound for index 1 (integer)
5. Lower bound for index 2 (integer)
6. Upper bound for index 2 (integer)
7. Lower bound for index 3 (integer)
8. Upper bound for index 3 (integer)
9. Lower bound for index 4 (integer)
10. Upper bound for index 4 (integer)
11. Lower bound for index 5 (integer)
12. Upper bound for index 5 (integer)
13. Lower bound for index 6 (integer)
14. Upper bound for index 6 (integer)

The index bounds are as described above for the ARRAY format. They can be defaulted by ending the record prematurely with a slash, or individually in the same way as \(<\text{missing}>\) values are declared. The operator cannot be defaulted.

Example

```
--operator--value-----I1--I2---J1--J2---K1--K2---
PORO S
  '=' 0.15  2  3  1  3  4  4  /
  '+' 0.03  2  2  2  3  4  4  /
  '=' 0.11  3  1*  3  1*  3  1*  /
/
```
Introduction

Under certain conditions, the Basic Interpolation Algorithm gives the same results as bilinear interpolation.

More precisely, suppose the reservoir grid lines in the i direction are all parallel, and the grid lines in the j direction are all parallel. Suppose there are just 4 data points, at the 4 corners of the grid. Then the interpolated values using the Basic Interpolation Algorithm are the same as calculated by bilinear interpolation along the grid lines.

Consider the inverse distance interpolation between values \( v_1 \) and \( v_2 \), at points \( p_1 \) and \( p_2 \), along a straight line, as shown in Figure B.1.

**Figure B.1** Inverse distance interpolation along a straight line

\[
\begin{align*}
&v_1 & d_1 & v & d_2 & v_2 \\
&p_1 & p & p & p_2
\end{align*}
\]

The interpolated value \( v \) at point \( p \) on the line is

\[
v = \frac{(1/d_1)v_1 + (1/d_2)v_2}{(1/d_1) + (1/d_2)} \quad \text{[EQ B.1]}
\]

\[
v = \frac{d_2v_1 + d_1v_2}{d_2 + d_1}
\]

which is just the linearly interpolated value.

Using this, and the fact that the distances along the grid lines in the i direction are independent of j, and similarly for the other grid lines, it can be shown that the Basic Interpolation Algorithm gives the same results as bilinear interpolation.
More generally, with the same condition on the x and y coordinates but with an arbitrary
distribution of data values, the same result applies within any rectangle of the interpolation grid
which is bounded by lines of the skeleton. (This is a parallelogram on the reservoir grid.) In
other words, it applies to those points of the interpolation grid which do not lie on the skeleton.
The distance formula used in interpolation is $d = \sqrt{x^2 + y^2}$ even in cylindrical coordinates, where $x = r$, $y = \theta$.

- Make sure that interpolation is independent in $x$ and $y$, by supplying data values at all intersection points of a sparse grid, as shown:

```
* * * X * * X * * * * * X
* * * X * * X * * * * * X
* * * * * * * * * * * *
* * * X * * X * * * * * X
* * * * * * * * * * * *
* * * * * * * * * * * *
```

The interpolation procedure does not use derivatives. This means that extrapolation is constant extrapolation from the last value available.

- Provide data values at the extremities of the layer.

With sloping faults, $z$ values supplied at a fault coordinate line for the top and bottom corners of a block will naturally refer to different $(x, y)$ locations. Consequently, the difference between the $z$ values will not represent the vertical thickness of the block at an $(x, y)$ location.

- Specify the $z$ and $dz$ values taking the slope into account.

No consistency checking is carried out on $x$ or $y$ coordinates. There no warning about $(x, y)$ grids where a cell intersects itself (like a bow tie) or two columns of cells intersect.

- ECLIPSE has data checks for $x$, $y$ and $z$. Alternatively, the grid graphics may be used to verify the data, using sections $x$-$y$, $y$-$z$ and $x$-$z$. 
Sample Problems

Key to sample problems

1. The interpolation of porosity data for a layer.
   This example is used in the section on the FILL operation to illustrate the interpolation algorithms.

2. Corner point geometry with sloping faults.
   This problem is discussed in the section on defining the reservoir grid. The simpler reservoir examples there are easily constructed from this input file by deleting unwanted statements.

3. Corner point geometry with distorted grid.
   This problem is discussed in the same section under the heading “A Distorted Areal Grid”.

4. FILL applied to x and y coordinates.
   This illustrates on a small scale the way in which x and y coordinates may be manipulated, to construct a grid which is fairly regular, but which has irregular features. Typically this is done to represent an aquifer. The grid is illustrated in Figure 2.7.

5. Conversion from block-centre to corner point geometry: flat block method. This shows the use of a mixture of block-centre and corner point geometry. It also shows the use of multiple reservoirs allowing change of coordinates with depth. A section through the field is shown in Figure 3.4. If the FLATBLCK keyword is replaced by the SLOPBLCK keyword, then the sloping block method is used - this is illustrated in Figure 3.5.

6. Output controls.
   This problem shows how to prepare a file containing just interpolated porosity data, by suppressing the output of ZCORN and COORD etc. using the OUTFILL keyword. Only the porosities at well positions are available: the other porosities are estimated from these.

7. Radial Grid example.
Interpolation of porosity (1)

Reservoir number 1 has 5 blocks in the x-direction, 4 blocks in the y-direction and 1 block in the z-direction. The radial option is set to false (F) so the grid is cartesian.

```
SPECGRID
  5 4 1 1 F /
```

The regular orthogonal grid has 1000 ft blocks in both the x and y directions. This defines x and y at the tops of the coordinate lines.

```
DXV
  5*1000 /
DYV
  4*1000 /
```

Depths and thicknesses.

```
TOPSNODE S
  '==' 1000.0 /
/
DZNODE S
  '==' 10 /
/
```

Specify the known porosity values.

```
PORO
  1* 0.1 1* 1* 0.2
  1* 1* 1* 0.3 1*
  1* 1* 1* 1* 1*
  1* 1* 0.4 1* 0.05 /
```

Perform interpolation.

```
FILL
```

Suppress unwanted output.

```
OUTFILL
  'ALL' 'NO' /
  'PORO' 'YES' /
/
```

Optional end of data keyword.

```
END
```
Corner point geometry with sloping faults (2)

Get a progress log on the screen.

```
OUTPUTS
  'logterm' 1 /
  'gridfile' 1 /
```

Reservoir number 1 has 4 blocks in the x-direction, 4 blocks in the y-direction and 4 blocks in the z-direction. The radial option is set to false (F) so the grid is cartesian.

```
SPECGRID
  4 4 4 1 F /
```

The regular orthogonal grid has 4100 ft blocks in both the x and y directions. This defines x and y at the tops of the coordinate lines.

```
DXV
  4*1000 /
DYV
  4*1000 /
```

Set the depths of nodes at the top of the top layer and at the top of the third layer. TOPSNODE data goes straight into ZCORN storage.

```
TOPSNODE
  5000 3* 5100 15* 5400 3* 5420
  25*
  5060 5067 18* 5460 5447 3*
  25* /
```

Fill in (x,y) data in COORD by summing x·s with DXV and y·s with DYV. Fill out the depths of nodes at the top of the top layer and at the top of the third layer. Supply depths for the tops and bottoms of coordinate lines (in COORD) from the top surface of the reservoir (in ZCORN).

```
FILL
```

Make missing the tops of the third layer of blocks which must be calculated from D2NODE data: as depths are taken from the layer above in preference to interpolating. The relevant blocks have node values in the range

- IN = 3 to 5
- JN = 1 to 5
- KN = 3 to 3

```
TOPSNODE S
  '='*, 1* , 3 5 , 1 5 , 3 3 /
```
Specify the reservoir thickness at nodes.

```
DZNODE
  20 2* 8 16* 20 2* 8 1*
  20 2* 8 16* 20 2* 8 1*
  20 2* 8 16* 20 2* 8 1*
  20 2* 8 16* 20 2* 8 1* /
```

Porosity data.

```
PORO
  16*0.2
  0.22 2* 0.16 8* 0.1 2* 0.31
  16*0.25
  16*0.1 /
```

Fill out the depths and porosities.

```
FILL
```

Depress the fault region by a vertical distance of 10 ft. The fault block range is IB = 4, JB = 2 TO 4, all KB

The ZCORN format is

```
-- OP VALUE L/R IB RANGE B/F JBRANGE T/B KBRANGE
ZCORN S
  '+', 10.0 2* 4 4 2* 2 4 2* 2* /
/
```

At this stage the depths of the tops and bottoms of the coordinate lines have defaulted to the top of the reservoir. To introduce a 30 degree sloping fault in the x-direction we first move the bottoms of the coordinate lines down 173.2 ft and then 100 ft in the x-direction. The relevant coordinate lines are in the range IN = 4 and JN = 2 to 5 The COORD data format is

```
-- OP DIST DIR T/B IN RANGE JN RANGE
COORD S
  '+', 173.2, 3 3 , 2 2 /
  '+', 100.0, 1 1 , 2 2 , 4 4 , 2 5 /
/
```

Modify the porosities with block indices IB, JB and KB in the range 2 to 3.

```
-- OP MODIFIER IB RANGE JBRANGE KBRANGE
PORO S
  '+', 0.1 2 3 2 3 2 3 /
  '*', 0.9 2 3 2 3 2 3 /
/
END
```
Corner point geometry with distorted grid (3)

Reservoir number 1 has 4 blocks in the x-direction, 4 blocks in the y-direction and 4 blocks in the z-direction. The radial option is set to false (F) so the grid is cartesian.

```
SPECGRID
4 4 4 1 F /
```

The COORD keyword is used in ARRAY format to specify pairs of x and y coordinates for the tops of the coordinate lines.

```
--OPERATOR DIMENSIONS TOP/BOTTOM
-- SET X,Y TOP ONLY
COORD A
  '=' 1,2 1,1 /
  1 1, 2 0, 4 0, 6 0, 6.5 0.5
  0 2, 3 3, 4 3, 5 3, 7 2
  1 4, 3 4, 4 4, 5 4, 7.5 4
  1 5, 3 5, 4 5, 5 5, 8 6
  1 6, 3 6, 4 7, 6 8, 8 9 /

COORD S
  '*' 1000.0 /
/
```

Apply a scale factor to scale up to field dimensions.

```
TOPSNODE
  '25*5000 75*
/
```

Specify depths of the nodes at the top of the reservoir.

```
DZNODE
  100*20 /
```

All four layers have the same thickness.

Default the x and y coordinates of coordinate lines (in COORD) copy the (x,y) coordinates from the top to the bottom. Compute the node depths. Fill in the depths of the top and bottom points on the coordinate lines, using the depths of the reservoir.

```
FILL
```
The **FILL** operation applied to x coordinates (4)

This example demonstrates on a small scale a facility available for complex reservoir grids. Set a convenient level of monitoring, and select grid geometry file.

\begin{verbatim}
OUTPUTS
 'LOGTERM' 1 /
 'LOGPRINT' 2 /
 'GRIDFILE' 1 /
 /
SPECGRID
  5 4 1 1 F /
/

Reservoir number 1 has 5 blocks in the x-direction, 4 blocks in the y-direction and 1 block in the z-direction. The radial option is set to false (F) so the grid is cartesian.

\begin{verbatim}
DXV 5*100 /
DYV 4*100 /
/

The blocks have an extent of 100 ft in the x direction, where not overridden by the explicit x values set below for COORD. The blocks are 100 ft in the y direction.

\begin{verbatim}
COORD A
  '=' 1 l, 1 l /
    1* 1* 1* 700 950 1* 1* 1* 700 950 1* /
    1* 1* 1* 600 1100 1* 1* 1* 600 1100 1* /
/

Set up some dummy depths and thicknesses etc.

\end{verbatim}

\begin{verbatim}
TOPSNODE S
  '=' 1000.0 /
/
DZNODE S
  '=' 100.0 /
/

Set up the coordinates.

\end{verbatim}

\begin{verbatim}
FILL
END
\end{verbatim}
\end{verbatim}
Conversion of block-centre depths (**TOPS, DEPTH, DZ**) (5)

Flat block option.
Set a convenient level of monitoring, and select grid geometry file.

```
OUTPUTS
'LOGTERM' 1 /
'GRIDFILE' 1 /
/
```

There are two reservoirs. Each is 5 blocks in the x direction and 3 blocks in the y direction. Reservoir 1 extends from blocks 1 to 2 in the z direction; reservoir 2 extends from blocks 3 to 4. There are 4 blocks in total in the z direction.

```
SPECGRID
  5 3 4 2 /
COORDSYS
  1 2 /
         3 4 /
```

Each reservoir is initially given the same regular (x,y) grid, with irregular spacing.

```
DXV
  1000.0 2000.0 1500.0 1000.0 800.0
  1000.0 2000.0 1500.0 1000.0 800.0 /
DYV
  1500.0 1000.0 2000.0
  1500.0 1000.0 2000.0 /
```

The grid points (3,2), (4,2) and (2,3) in reservoir 1 are assigned x and y values, for the top of the coordinate lines.

```
COORD A
  '=" 1 2, 1 1, 3 3, 2 2, 1 1 /
    3000.0 1500.0 /
COORD A
  '=" 1 2, 1 1, 4 4, 2 2, 1 1 /
    4000.0 2000.0 /
COORD A
  '=" 1 2, 1 1, 2 2, 3 3, 1 1 /
    1500.0 2500.0 /
```

Set the top-centre depths for the 4 corner blocks of layer 1.

```
TOPS S
  '=" 500.0 1 1*, 1 1*, 1 1* /
  '=" 700.0 5 1*, 1 1*, 1 1* /
  '=" 800.0 1 1*, 3 1*, 1 1* /
  '=" 1000.0 5 1*, 3 1*, 1 1* /
/
```
Set the block-centre depths for the 4 corner blocks of layer 3.

Z S
=' 1300.0 1 1*, 1 1*, 3 1* /
=' 1500.0 5 1*, 1 1*, 3 1* /
=' 1400.0 1 1*, 3 1*, 3 1* /
=' 1700.0 5 1*, 3 1*, 3 1* /
/

Make each layer constant thickness.

DZ A
=' 1 1, 1 1 /
150.0
250.0
100.0
200.0 /

Set the flat-block option for conversion to corner point geometry, making the resulting blocks flat-topped.

FLATBLCK

Set up the x and y coordinates, perform the conversions, and sum the thicknesses to produce ZCORN.

FILL

Modify the coordinate lines for reservoir 1. Make all the coordinate lines 1000 ft high, by moving the bottom points down 1000 ft. Then make the coordinate lines slope, for nodes 3 to 6 in the x direction and nodes 3 to 4 in the y direction, by moving the bottom points 1000 ft in the x and y directions.

COORD S
='+ 1000.0 3 3, 2 2, 4* 1 1 /
='+ 1000.0 1 2, 2 2, 3 6, 3 4, 1 1 /
/
END
Output controls to suppress ZCORN etc. (6)

The reservoir has 19 blocks in the x direction, 11 blocks in the y direction and 1 block in the z direction. The reservoir grid is cartesian by default.

```
SPECGRID
19 11 1 /
```

A rectangular (x,y) grid with irregular spacing is used.

```
DXV
1968 11*1574 7*1968 /
DYV
2230 10*1837 /
```

The porosity data. This might typically come from well data, where the wells are at locations (8,7), (13,7), and so on.

```
PORO S
'=' .15 8 8 7 7 /
'=' .126 13 13 7 7 /
'=' .126 4 4 5 5 /
'=' .14 9 9 5 5 /
'=' .147 11 11 6 6 /
'=' .120 10 10 8 8 /
/
```

Set up the coordinates from dx and dy, and perform the interpolation.

```
FILL
```

Suppress output for everything except porosity.

```
OUTFILL
'PORO' 'YES' /
'ACTNUM' 'NO' /
'COORD' 'NO' /
'ZCORN' 'NO' /
'SPECGRID' 'NO' /
'COORDSYS' 'NO' /
/
END
```
Radial data (7)

The reservoir has 19 blocks in the r direction, 11 blocks in the theta direction and 2 blocks in the z direction. The reservoir grid is radial.

Specgrid

```
OUTPUTS
  'LOGTERM' 4 /
  'LOGPRINT' 4 /
  'GRIDFILE' 4 /
/
DEBUG
  'LOGGING' 4 /
  'STORAGE' 4 /
/

SPECGRID
  19 11 2 1 T /
RADV
  .125 .25 .5 .1 .2 .4 .8 1.6 3.2 6.4 12.8 25.6 51.2 102.4 204.8
  409.6 819.2 1638.4 3276.8 6353.6 /
DTHETAV
  60 10*30 /
TOPS
  100 207* 200 110 207* 210 /
FILL
DZ
  209*10 209*20 /
```
The porosity data. This might typically come from well data, where the wells are at locations
(8,7), (13,7), and so on.

```
PORO S
'=' .15 8 8 7 7 /
'=' .126 13 13 7 7 /
'=' .126 4 4 5 5 /
'=' .14 9 9 5 5 /
'=' .147 11 11 6 6 /
'=' .120 10 10 8 8 /
/
PERMR
10* 100 3* 200 3* 150
171*
10* 200 3* 200 3* 250
100 207* 100
/
PERMTHT
100 2* 120 2* 140 1* 150
191*
100 2* 120 2* 140 1* 150
150 207* 160 /
PERMZ
10 416* 20 /
MULTR
6*1 0 3* 9*1
171*
5*1 1* 0 3* 9*1
6*1 0 3* 9*1
171*
5*1 1* 0 3* 9*1 /
MULTTHT
9*1 1* .5 1* 406*1 /
MULTZ
418*.9 /
```

Set up the coordinates from dx and dy, and perform the interpolation.

```
FILL
```

Suppress output for everything except porosity.

```
OUTFILL
'PORO' 'YES' /
'ACTNUM' 'NO' /
'COORD' 'NO' /
'ZCORN' 'NO' /
'SPECGRID' 'NO' /
'COORDSYS' 'NO' /
/
END
```
Index

Appendix E

A
Active Grid Block Region . 61
ACTNUM . . . . . . . . . . . . . . . 61
Areal Grid
Distorted . . . . . . . . . . 27

B
Bilinear Interpolation . . . . 127
Block-centre depths
Conversion . . . . . . . . 137

C
COORD . . . . . . . . . . . . . . . . 63
Coordinates
Grid Block . . . . . . . . . . 56
Lines . . . . . . . . . . . . . . 63
Node . . . . . . . . . . . . . . 56
System Information . . . . 64
COORDSYS . . . . . . . . . . . . 64
Corner Point Geometry
Distorted Grid . . . . 135
Simple Grid . . . . . . . . 16
Sloping Faults . . . . 133

D
Data Formats . . . . . . . . . . 50
ARRAY . . . . . . . . . . . . 122
ECLIPSE . . . . . . . . . . . . 124
SINGLE value list . . . . 125
DEBUG . . . . . . . . . . . . . . . . 65
Defining the Reservoir Grid 11
DEPTH . . . . . . . . . . . . 66
DTHETAV . . . . . . . . . . . . 67
DXV . . . . . . . . . . . . . . . . . . 68
DYV . . . . . . . . . . . . . . . . . . 69
dz . . . . . . . . . . . . . . . . . . . . 70
dzCORN . . . . . . . . . . . . . . . . 71
dzNODE . . . . . . . . . . . . . . . . 72

E
END . . . . . . . . . . . . . . . . . . . . 73
End of Input Data . . . . . . . . . 73

F
Fault
Sloping . . . . . . . . . . . . 23
Vertical . . . . . . . . . . . . 21
FILL . . . . . . . . . . . . . . . . . . . . 74

Fill
How Fill Works . . . . . . . . 35
Introduction . . . . . . . . . 9, 35
Outline . . . . . . . . . . . . 36
Output . . . . . . . . . . . . 32
Filling-in Missing Values 17, 74
Flat Block Mode for Geometry
Conversion . . . . . . . . . . . . 75
FLATBLCK . . . . . . . . . . . . . . 75

G
Gap . . . . . . . . . . . . . . . . . . . 19
Geometry Conversion . . . . 29
Grid
Characteristics . . . . . . . . . 103
Definition . . . . . . . . . . . . 11
Introduction . . . . . . . . . . . 11
Simple . . . . . . . . . . . . . . 13
Grid Axes
Map Coordinates . . . . 77
Grid Block
Centre Depths . . . . . . . . 66, 117
Corner Depths . . . . . . . . 118
Pore Volumes . . . . . . . . 100
Porosities . . . . . . . . . . . . 99
Top Centre Depths . . . 105
Grid Block Sizes
Angular . . . . . . . . . . . . 67
X Direction . . . . . . . . 68
Y Direction . . . . . . . . 69
Z Direction . . . . . . . . .70
Z Direction at Block Edges 71
Z Direction at Nodes . . .72

Grid Nodes
Block Top Depths . .106
Depths . . . . . . . . . .119
R Coordinates . .101
Theta Coordinates . .104
X Coordinates . . . . .115
Y Coordinates . . . . .116

N
Name of data file to be included 76
Net to Gross Ratios . . . . .87
NTG . . . . . . . . . . . .87

O
OUTFILL . . . . . . . . .88
Outline . . . . . . . . . .36
OUTPUT . . . . . . . . .89
Output Control . . . . . 89
Debug Stream . . . . .65
FILLED Stream . . . . .88
ZCORN Suppression . 139

P
Permeabilities
I Direction . . . . . .91
J Direction . . . . . .92
K Direction . . . . . .93
R Direction . . . . . .94
Theta Direction . . . .95
X Direction . . . . . .96
Y Direction . . . . . .97
Z Direction . . . . . .98
PERMI . . . . . . . . .91
PERMJ . . . . . . . . .92
PERMK . . . . . . . . .93
PERMR . . . . . . . . .94
PERMTHT . . . . . .95
PERMX . . . . . . . . .96
PERMY . . . . . . . . .97
PERMZ . . . . . . . . .98
PORO . . . . . . . . . .99
PORV . . . . . . . . .100

R
Radial Data . . . . . .140
RADV . . . . . . . . .101
Reset message print and stop limits 78

S
Sample Problems . . . .131
Key . . . . . . . . . . .131
SLOPBLCK . . . . . .102
Sloping Block Mode for Geometry Conversion . .102
SPECGRID . . . . . .103

T
THETAV . . . . . . . . .104
TOPS . . . . . . . . . . .105
TOPSNOKE . . . . .106
TRAN . . . . . . . . . .113
TRANI . . . . . . . . .107
TRANJ . . . . . . . . .108
TRANK . . . . . . . . .109
TRANR . . . . . . . . .110
Transmissibilities
I Direction . . . . .107
J Direction . . . . .108
K Direction . . . . .109
R Direction . . . . .110
Theta Direction . . .111
X Direction . . . . .112
Z Direction . . . . .114
Transmissibility Multipliers
I Direction . . . . .79
J Direction . . . . .80
K Direction . . . . .81
R Direction . . . . .82
Theta Direction . . .83
X Direction . . . . .84
Y Direction . . . . .85
Z Direction . . . . .86
TRANTH . . . . . .111
TRANX . . . . . .112
TRANZ . . . . . .114

U
Using Fill . . . . . .33
W
Wedge . . . . . . . . . . . . . . . 19

X
X coordinates

FILL operation . . . . 136
XV . . . . . . . . . . . . . . . 115

Y
YV . . . . . . . . . . . . . . . 116

Z
Z . . . . . . . . . . . . . . . . 117
ZCORN . . . . . . . . . . . . . 118
ZNODE . . . . . . . . . . . . . 119