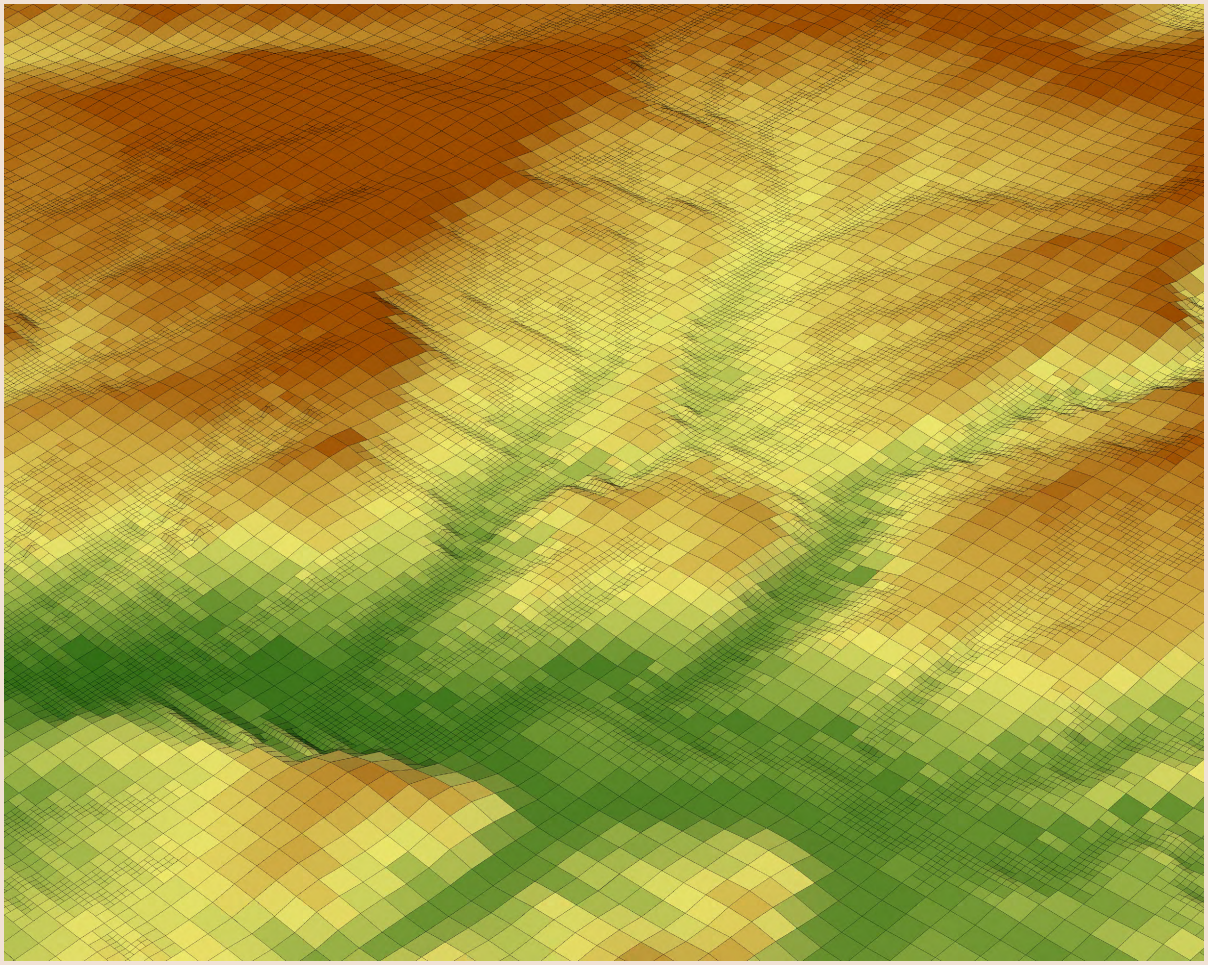
USGS
science for a changing world

Groundwater Resources Program

Prepared in cooperation with George Mason University

GRIDGEN Version 1.0: A Computer Program for Generating Unstructured Finite-Volume Grids



Open-File Report 2014–1109

U.S. Department of the Interior
U.S. Geological Survey

Groundwater Resources Program
Prepared in cooperation with George Mason University

GRIDGEN Version 1.0: A Computer Program for Generating Unstructured Finite-Volume Grids

By Jyh-Ming Lien, Guilin Liu, and Christian D. Langevin

Open-File Report 2014–1109

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
SALLY JEWELL, Secretary

U.S. Geological Survey
Suzette M. Kimball, Acting Director

U.S. Geological Survey, Reston, Virginia: 2015

Suggested Citation
Lien, Jyh-Ming, Liu, Guilin, and Langevin, C.D., 2015, GRIDGEN version 1.0—A computer program for generating unstructured finite-volume grids: U.S. Geological Survey Open-File Report 2014–1109, 26 p., *http://dx.doi.org10.3133/ofr20141109*.

ISSN 2331-1258 (online)

Acknowledgments

Development of the GRIDGEN computer program was supported by the U.S. Geological Survey (USGS) Groundwater Resources Program. The authors are grateful to USGS employees Scott Paulinski and Brian Clark for thoughtful and constructive reviews of the GRIDGEN software and this report. The authors are also grateful to Andy Leaf and Daniel Feinstein for testing earlier versions of the program.

Contents

Figures

Tables

GRIDGEN Version 1.0: A Computer Program for Generating Unstructured Finite-Volume Grids

By Jyh-Ming Lien,[1] Guilin Liu,[1] and Christian D. Langevin[2]

Abstract

GRIDGEN is a computer program for creating layered quadtree grids for use with numerical models, such as the MODFLOW–USG program for simulation of groundwater flow. The program begins by reading a three-dimensional base grid, which can have variable row and column widths and spatially variable cell top and bottom elevations. From this base grid, GRIDGEN will continuously divide into four any cell intersecting user-provided refinement features (points, lines, and polygons) until the desired level of refinement is reached. GRIDGEN will then smooth, or balance, the grid so that no two adjacent cells, including overlying and underlying cells, differ by more than a user-specified level tolerance. Once these gridding processes are completed, GRIDGEN saves a tree structure file so that the layered quadtree grid can be quickly reconstructed as needed. Once a tree structure file has been created, GRIDGEN can then be used to (1) export the layered quadtree grid as a shapefile, (2) export grid connectivity and cell information as ASCII text files for use with MODFLOW–USG or other numerical models, and (3) intersect the grid with shapefiles of points, lines, or polygons, and save intersection output as ASCII text files and shapefiles. The GRIDGEN program is demonstrated by creating a layered quadtree grid for the Biscayne aquifer in Miami-Dade County, Florida, using hydrologic features to control where refinement is added.

Introduction

An unstructured grid version of MODFLOW, called MODFLOW–USG (Panday and others, 2013) simulates groundwater flow using a generic finite-volume approach. This flexibility allows grids, other than the structured finite-difference grid required by previous MODFLOW versions, to be used to discretize the model domain. Use of an unstructured model grid offers many advantages, including the capability to refine areas of interest with additional levels of grid resolution. Design and manipulation of unstructured model grids, however, is more complicated than working with structured finite-difference grids. Most of the added complexity stems from the requirement that information about how cells are connected to one another must be provided by the user. With a structured finite-difference grid, cell connections are an intrinsic grid property in that each cell is connected to the six adjacent cells, and identification of the six connected cells can be easily determined.

This report describes a new computer program called GRIDGEN, which can be used to generate an unstructured finite-volume model grid. This first GRIDGEN version (Version 1.0) is designed to

[1]George Mason University.
[2]U.S. Geological Survey.

support the design and manipulation of a three-dimensional, layered quadtree grid. A layered quadtree grid is an enhancement to the traditional MODFLOW grid, in that any cell can be divided into four equal-sized cells. It is called a "layered" quadtree grid because it has the same number of layers as the base MODFLOW grid from which it was derived. The primary advantage of a layered quadtree grid is that additional levels of resolution can be added anywhere within the model domain to better represent groundwater flow, for example, near wells and streams. Layered quadtree grids also share important similarities with traditional MODFLOW grids, such as rectangular cells and three principal (and orthogonal) grid directions.

GRIDGEN is a grid generation tool designed primarily for MODFLOW–USG models; however, it may be useful for generating grids for other numerical models. GRIDGEN is a command-line executable program that reads and writes ESRI (1998) shapefiles. It is not a graphical user interface, and it has no post-processing capabilities. GRIDGEN creates most of the discretization information required by MODFLOW–USG, including the cell connectivity information and other cell connection properties, but it does not create the unstructured discretization input file for MODFLOW–USG. Users will need to create this file manually or through other methods.

This report describes GRIDGEN Version 1.0, beginning with an overview of the program, including the types of files that it reads and writes, followed by a description of the format for the different types of information blocks that GRIDGEN processes. An example is then provided showing how GRIDGEN can be used to create a layered quadtree grid, write information about the grid, create shapefiles of the grid, and then intersect the grid with spatial hydrologic features.

Overview of the GRIDGEN Program

The GRIDGEN program is provided as a command-line executable program; however, the source code is also contained within the distribution for those who would like to compile the program. The program is written in standard C++ and has been successfully compiled on Windows, Macintosh, and Linux operating systems. A C library called shapelib (Warmerdam, 1998) is used by GRIDGEN to read and write shapefiles and the associated attribute files (with the .shx and .dbf file extensions). Successful use of the GRIDGEN program will require familiarity with running programs from the command line.

Spatial Information

The GRIDGEN program reads and writes shapefiles, the format of which is described in detail by ESRI (1998). For the program to work as intended, the shapefiles must be valid and have all information in the same geographic units as the model grid (commonly feet or meters). GRIDGEN is capable of reading and writing several types of spatial features, including points, lines, and polygons.

Definition Files and Information Blocks

In addition to shapefiles, the GRIDGEN program reads and writes definition files, which are simple ASCII text files. Definition files are normally given the ".dfn" file suffix to differentiate them from other types of files. Definition files are used to provide instructions and other information required by GRIDGEN. Definition files are intended to be simple so that they can be manually created and understood by users. For this reason, definition files cannot store arrays of numbers and other long lists of information; where arrays of numbers are required, a definition file can point to a file containing the array.

Definition files contain one or more blocks of information. These blocks consist of keywords, which are recognized by the program and shown here in capital letters, and other information provided by the user. A block always starts with the "BEGIN" keyword and terminates with an "END" keyword. Immediately following the "BEGIN" and "END" keywords is the block type. On the first line of the block, following the block type, a unique name is required. The block name is a unique identifier that can be used in other blocks to refer to the named block.

The body of the block contains a list of records, which are in no particular order. For example, a generic block would have the following form:

```
BEGIN block_type block_name
  record_name_1 = record_value_1
  record_name_2 = record_value_2
  …
  record_name_n = record_value_n
END block_type
```

This block structure allows the user to provide named input to GRIDGEN and allows new input options and keywords to be added in a backward-compatible way as new functionality for the program is developed.

Record values are generally text strings, integer values, real values, or boolean flags (True or False). Record values are indicated in the block descriptions in this report using the following syntax:

```
record_name = <text>
record_name = <integer>
record_name = <real>
record_name = <boolean>
```

Definition files also support arrays, which are indicated by the array type followed by brackets and the array shape, as indicated by the following lines:

```
record_name = <integer[NCOL]>
record_name = <real[NROW,NCOL]>
```

There are two options for assigning array values, and they are patterned after two of the MODFLOW–2005 array reading utilities (Harbaugh, 2005). The first option is for assigning a constant value to the entire array. For this case, the CONSTANT keyword can be used. For example, the following line would assign a constant value of 50.0 to the DELR array:

```
DELR = CONSTANT 50.0
```

For an array with non-constant values, the array values must be listed in an external array file, which is simply an ASCII text file containing all of the array values. The array values must be listed in row-major order, as is done for MODFLOW–2005. The syntax for assigning an array using values from an external file is

```
DELR = OPEN/CLOSE delr.dat
```

Within GRIDGEN definition files, there are two general functions of blocks, namely to define objects and to define actions. An object block might describe a grid or a quadtree refinement feature, whereas an action block contains instructions that can be executed by the GRIDGEN program. Action blocks can be executed by a user by running the GRIDGEN program with the name of the action block

and the name of the definition file provided as arguments. For example, the following command-line statement will execute an action block called “qtgbuilder” in a definition file named “qtg.dfn”: “gridgen.exe qtgbuilder qtg.dfn”. It is also possible to execute the GRIDGEN program with an object block. In some cases, the program will perform internal calculations in order to create a representation of the object, but no output will be generated.

GRIDGEN Definition Blocks

The block types supported in GRIDGEN Version 1.0 are listed in Table 1. Although most of these blocks would be used in a typical GRIDGEN application, not all of them are required in every instance. This report section provides the details for each of these blocks.

Table 1. Block types available in GRIDGEN Version 1.0.

Block type	Block function	Block description
MODFLOW_GRID	Define object	Contains the information required to characterize a three-dimensional MODFLOW grid.
QUADTREE	Define object	Contains the information required to characterize a three-dimensional layered quadtree grid.
QUADTREE_BUILDER	Define action	An action block that will create a QUADTREE grid
REFINEMENT_FEATURES	Define object	Features from a shapefile that are used by QUADTREE_BUILDER
ACTIVE_DOMAIN	Define object	A polygon feature from a shapefile that will be used by the QUADTREE_BUILDER block to specify the active domain of the grid.
GRID_TO_SHAPEFILE	Define action	An action block that will create a shapefile for the specified grid
GRID_TO_USGDATA	Define action	An action block that will write information about the cell connectivity and geometry of the specified grid.
GRID_TO_VTKFILE	Define action	An action block that will create a Visualization Toolkit (VTK) file for the specified grid.
GRID_INTERSECTION	Define action	An action block that will write information about the spatial intersection of features from a shapefile and the specified grid.

MODFLOW_GRID

The term “MODFLOW grid” is used here because it conforms to the grid type that can be used with the MODFLOW program for simulation of groundwater flow (Harbaugh, 2005). A MODFLOW grid consists of one or more layers, rows, and columns. The number of layers, rows, and columns are defined by the NLAY, NROW, and NCOL record identifiers, respectively. Columns and rows can have variable spacings within the grid and are thus characterized by one-dimensional arrays. Column spacings are contained within the DELR array (of size NCOL), and row spacings are contained in the DELC array (of size NROW). In plan view, rows increase in number from top to bottom and columns increase in number from left to right. Layers increase in number from top to bottom, and thus the top layer is layer 1. A top elevation is provided for every cell in layer 1. Bottom elevations are provided for every cell in every layer. With this design, layer thicknesses can vary across the grid in order to represent subsurface hydrogeologic units. Note that in a MODFLOW_GRID block, top elevations are only provided for the top layer; for the underlying layers, cell top elevations are equal to the bottom elevations of the overlying cells.

A MODFLOW grid is described in definition files by the MODFLOW_GRID block type. A complete description of the MODFLOW_GRID block is shown in table 2. A MODFLOW_GRID can be used for subsequent processing by other action blocks, but more importantly, it is required as a base

grid for building layered quadtree grids (described in the next section). The following lines demonstrate the use of the MODFLOW_GRID block to characterize a MODFLOW grid having 3 layers, 30 rows, and 40 columns, with each layer having a constant top and bottom elevation. The GRIDGEN program will see the name of this MODFLOW grid as "basegrid."

```
#comment – this is a MODFLOW_GRID with the name "basegrid"
BEGIN MODFLOW_GRID basegrid
  ROTATION_ANGLE = 15.
  X_OFFSET = 5000.
  Y_OFFSET = 5200.
  LENGTH_UNIT = undefined
  NLAY = 3
  NROW = 30
  NCOL = 40
  DELR = CONSTANT 100.0
  DELC = CONSTANT 100.0
  TOP = CONSTANT 75.0
  BOTTOM LAYER 1 = CONSTANT -100.0
  BOTTOM LAYER 2 = CONSTANT -110.0
  BOTTOM LAYER 3 = CONSTANT -275.0
END MODFLOW_GRID
```

Note that for this simple example, the one-dimensional arrays (DELR and DELC) and the two-dimensional arrays (TOP and BOTTOM) have all been assigned constant values. As previously described, non-constant arrays can also be used with GRIDGEN by referring to external ASCII text files that contain the array values.

Table 2. List of the records that compose a MODFLOW_GRID block.
[NCOL, number of columns; NROW, number of rows; NLAY, number of layers]

Record name	Numerical type	Default	Record description
ROTATION_ANGLE	<real>	Required	Clockwise rotation angle of upper left corner, in degrees
X_OFFSET	<real>	Required	x spatial coordinate of grid lower left corner
Y_OFFSET	<real>	Required	y spatial coordinate of grid lower left corner
LENGTH_UNIT	<text>	Required	Undefined, foot, or meter
NLAY	<integer>	Required	Number of layers
NROW	<integer>	Required	Number of rows
NCOL	<integer>	Required	Number of columns
DELR	<real[NCOL]>	Required	1D array of column widths along a row
DELC	<real[NROW]>	Required	1D array of row widths along a column
TOP	<real[NROW, NCOL]>	Required	2D array of top elevations for model layer 1
BOTTOM LAYER 1	<real[NROW, NCOL]>	Required	2D array of bottom elevations for model layer 1
BOTTOM LAYER 2	<real[NROW, NCOL]>	Required	2D array of bottom elevations for model layer 2
(Repeat for all layers)	<real[NROW, NCOL]>	Required	Bottom elevations for other model layers (not shown)
BOTTOM LAYER NLAY	<real[NROW, NCOL]>	Required	2D array of bottom elevations for bottom model layer

QUADTREE

A layered quadtree grid is a rectilinear grid in which any cell can be divided into four equally sized cells. The term "quadtree" is used because the division of cells into groups of four results in a hierarchical tree structure. "Leaves" from the tree are then used to construct the grid for groundwater flow simulation. For the GRIDGEN program, a MODFLOW_GRID object (referred to herein as a base

grid) provides the foundation from which quadtree division takes place; thus, each cell in the MODFLOW_GRID has its own tree structure. The term “layered” is used to describe this particular quadtree grid implementation because the quadtree refinement pattern may be different for each layer of the base grid. The requirement of an underlying three-dimensional base grid is not commonly encountered in most quadtree implementations and should be recognized when evaluating whether or not GRIDGEN can be used for other applications.

An example of a simple quadtree grid is shown in figure 1. In this two-dimensional example, which starts from a 2-row and 2-column base grid, the cell in row 1 and column 2 was refined (divided into four) as were selected refined cells. Note that figure 1 shows only the leaves of the tree structure without showing the internal “branches” of the tree. An internal branch represents an intermediate level cell that has been further divided. Internal branches are important for defining the underlying tree structure, but they are not explicitly included as unique cells of the quadtree grid used for flow simulation. Only leaves from the tree are included as unique cells in the grid.

Cell numbering is required for programs like MODFLOW–USG, because cell numbers are used to describe cell connectivity and other grid properties. In GRIDGEN, cells are numbered recursively using row-major order. This means that cells contained in the first base grid cell (layer 1, column 1, row 1) are numbered first. Then the cells in the next base grid cell (layer 1, row 1, column 2) are numbered. This numbering proceeds first along columns, then along rows, and lastly for each layer. Within each base grid cell, quadrants are also numbered using row-major order, which means cells are numbered in the order of “northwest,” “northeast,” “southwest,” and “southeast.” This numbering scheme results in the cell numbers shown in figure 1. As described later, GRIDGEN also has the capability to exclude cells outside of the active domain from being numbered. Those cells are given a cell number of –1.

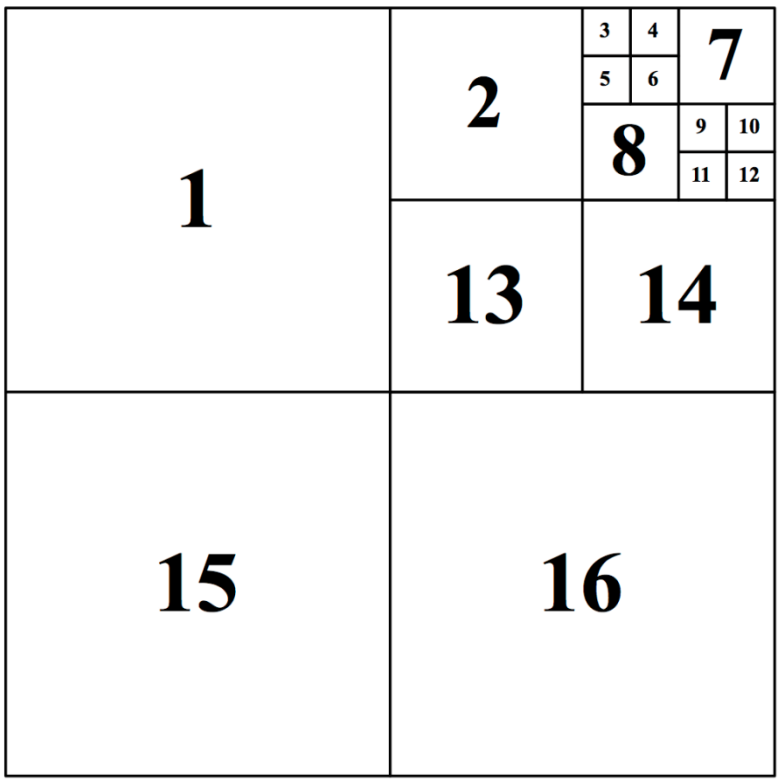


Figure 1. Diagram showing example of a quadtree grid.

Information on the structure of a layered quadtree grid is stored in a tree structure file, identified with the STRUCTURE_FILE keyword. As described later, the QUADTREE_BUILDER block can be used to create a tree structure file. For the quadtree grid shown in figure 1, the tree structure file contains the following 17 lines:

```
16
1, (1,1,1)
2, (1,1,2) 1
3, (1,1,2) 211
4, (1,1,2) 212
5, (1,1,2) 213
6, (1,1,2) 214
7, (1,1,2) 22
8, (1,1,2) 23
9, (1,1,2) 241
10, (1,1,2) 242
11, (1,1,2) 243
12, (1,1,2) 244
13, (1,1,2) 3
14, (1,1,2) 4
15, (1,2,1)
16, (1,2,2)
```

The first line contains the total number of cells, including those cells that may be assigned a –1 cell number to indicate they are excluded from the groundwater flow simulation. This total number of cells is equal to the number of leaves in the tree. For the grid in figure 1, there are 16 cells. The tree structure file then contains one line for each cell, which contains the cell identifier. The cell identifier is a text string consisting of the (layer, row, column) identifier of the cell within the base grid, followed by the quadrant sequence. The quadrant sequence is a list of quadrant numbers. Quadrant numbering follows the row-major ordering scheme in which 1 denotes the northwest quadrant, 2 denotes the northeast quadrant, 3 denotes the southwest quadrant, and 4 denotes the southeast quadrant. These quadrant numbers are given for each level of refinement until the cell position within the base grid cell is exactly described. Thus, the level of refinement for any cell is equal to the number of digits composing the quadrant sequence. The absence of a quadrant sequence for a cell indicates that the cell is a base grid cell having no refinement (a refinement level of zero).

Layered quadtree grids are described within a definition file using the QUADTREE block type. A QUADTREE block requires that a named MODFLOW_GRID be available. The MODFLOW_GRID block can be stored within the same definition file as the quadtree grid, or it can be accessed from within another definition file. An example of the definition file required to describe the simple quadtree grid in figure 1 is

```
#comment – this is the basegrid for the quadtree grid
BEGIN MODFLOW_GRID basegrid
  ROTATION_ANGLE = 0.
  X_OFFSET = 0
  Y_OFFSET = 0
  LENGTH_UNIT = undefined
  NLAY = 1
  NROW = 2
  NCOL = 2
  DELR = CONSTANT 1.0
  DELC = CONSTANT 1.0
```

```
  TOP = CONSTANT 1.0
  BOTTOM LAYER 1 = CONSTANT 0.0
END MODFLOW_GRID

#comment – this is the block describing the quadtree grid
BEGIN QUADTREE quadtreegrid
  MODFLOW_GRID = basegrid
  STRUCTURE_FILE = OPEN/CLOSE quadtreegrid.tsf
  TOP LAYER 1 = OPEN/CLOSE quadtreegrid.top1.dat
  BOTTOM LAYER 1 = OPEN/CLOSE quadtreegrid.bot1.dat
END QUADTREE
```

Note that the QUADTREE block itself is relatively short; it contains a MODFLOW_GRID record that refers to the name of the base MODFLOW grid. The block also references the STRUCTURE_FILE and top and bottom elevations for each layer. Note that both top and bottom elevations are required for each layer, which is different from a MODFLOW_GRID, because there may be a different number of cells within each layer of a quadtree grid.

Table 3 lists the records that compose the QUADTREE grid block.

Table 3. List of the records that compose the QUADTREE block.

Record name	Record type	Default	Record description
MODFLOW_GRID	<modflow_grid>	Required	MODFLOW_GRID block name
STRUCTURE_FILE	<text>	Required	Name of tree structure file
TOP LAYER 1	<real[number of cells in layer 1]>	Required	1D array of top elevations for model layer 1
TOP LAYER 2	<real[number of cells in layer 2]>	Required	1D array of top elevations for model layer 2
…		Required	Top elevations for other model layers (not shown)
TOP LAYER NLAY	<real[number of cells in layer NLAY]>	Required	1D array of top elevations for bottom model layer
BOTTOM LAYER 1	<real[number of cells in layer 1]>	Required	1D array of bottom elevations for model layer 1
BOTTOM LAYER 2	<real[number of cells in layer 2]>	Required	1D array of bottom elevations for model layer 2
…	<real[number of cells in layer]>	Required	Bottom elevations for other model layers (not shown)
BOTTOM LAYER NLAY	<real[number of cells in layer NLAY]>	Required	1D array of bottom elevations for bottom model layer

QUADTREE_BUILDER

Unlike a MODFLOW_GRID, in which the information is relatively simple and can be entered by hand into a definition file, quadtree grids are more complicated and require storing information about the tree structure. A powerful feature of the GRIDGEN program is the capability to create a quadtree grid using spatial features to control where refinement is added and where cells are active. Quadtree grids can be generated by creating and executing the QUADTREE_BUILDER block. Use of the QUADTREE_BUILDER block will be required as a first step in most instances where a quadtree grid is needed, unless a tree structure file can be created using an alternative method.

The following is an example of a QUADTREE_BUILDER block that can be used to construct a layered quadtree grid:

```
BEGIN QUADTREE_BUILDER quadtreebuilder
  MODFLOW_GRID = basegrid
  ACTIVE_DOMAIN LAYER 1 = active_domain_layer_1
  ACTIVE_DOMAIN LAYER 2 = active_domain_layer_2
  ACTIVE_DOMAIN LAYER 3 = active_domain_layer_3
  REFINEMENT_FEATURES LAYER 1 = ad_layer_1 river wells_layer_1
  REFINEMENT_FEATURES LAYER 3 = wells_layer_3
  SMOOTHING = full
  SMOOTHING_LEVEL_HORIZONTAL = 1
  SMOOTHING_LEVEL_VERTICAL = 1
  TOP LAYER 1= REPLICATE basegrid
  TOP LAYER 2= REPLICATE basegrid
  TOP LAYER 3= REPLICATE basegrid
  BOTTOM LAYER 1 = REPLICATE basegrid
  BOTTOM LAYER 2 = REPLICATE basegrid
  BOTTOM LAYER 3 = REPLICATE basegrid
  GRID_DEFINITION_FILE = quadtreegrid.dfn
END QUADTREE_BUILDER
```

The first record in the block refers to the base MODFLOW_GRID from which the quadtree grid is created. This example refers to a MODFLOW_GRID block named “basegrid,” which would need to be included in the file or made available through the LOAD keyword. The next lines in this example indicate the names of ACTIVE_DOMAIN blocks in which an ACTIVE_DOMAIN is specified for layers 1, 2, and 3. If an ACTIVE_DOMAIN is not specified for a layer, then the entire layer is active. REFINEMENT_ FEATURES can optionally be used for as many layers as necessary to refine the grid, and for each record entry, a list of REFINEMENT_FEATURES block names (separated by spaces) is included. Thus, multiple blocks of REFINEMENT_FEATURES can be specified for a layer. REFINEMENT_FEATURES do not have to be specified for every layer. In the QUADTREE_BUILDER example above, layer 1 is refined by three sets of features (contained in the ad_layer_1, river, and wells_layer_1 blocks) and layer 3 is refined by only one feature set (wells_layer_3).

Note that as part of the quadtree grid generation, the user has the option to smooth the refinement contrasts between adjacent cells. This is done in the program by continuously refining cells until the desired level of refinement contrast is met. In the example above, the smoothing operation is applied to the “full” grid (“none” would be the alternative to “full”) with SMOOTHING_LEVEL_HORIZONTAL = 1 and SMOOTHING_LEVEL_VERTICAL = 1, which means that the refinement level difference of two horizontally or vertically adjacent cells in the grid

would not exceed one. In figure 1, cells 2 and 14 would each be divided into four cells if SMOOTHING_LEVEL_HORIZONTAL was set to 1. Note that vertical smoothing can cause cells to be refined even if no REFINEMENT_FEATURES are specified for that layer. Specification of large smoothing level contrasts for these options will result in grids that are not smoothed.

Next in the example QUADTREE_BUILDER block are the TOP and BOTTOM assignments. These record assignments contain the REPLICATE keyword. This REPLICATE option indicates that the top or bottom elevations will be assigned by replicating base grid cell elevations. No interpolation is used with this option.

The INTERPOLATE option or the ASCIIGRID option could also have been used in the assignment of top and bottom elevations. If the INTERPOLATE option were used, then bilinear interpolation would have been used to calculate a top or bottom elevation for the center of each quadtree grid cell using top or bottom elevations from the base grid. Use of the ASCIIGRID option would have allowed an external elevation surface, as stored as an ASCII grid (ESRI, 2014), to be used for assignment of top and bottom elevations for each quadtree grid cell. With the ASCIIGRID option, an elevation value for the quadtree grid cell is calculated using an area-weighted average of all ASCII grid cells within the quadtree grid cell. Although GRIDGEN allows a model grid to be rotated in the x-y plane, there is no way to rotate an ASCII grid.

With the interpolation options for assigning top and bottom elevations to quadtree grid cells, the bottom of an overlying cell might not align with the top of an underlying cell, particularly if the level of refinement is different for the two cells. GRIDGEN contains an AUTOALIGNMENT option, whereby the bottom elevation of an overlying larger cell is automatically used as the top elevation of an underlying smaller cell. Likewise, the top elevation of an underlying larger cell is automatically assigned as the bottom elevation of an overlying smaller cell.

Interpolation and elevation errors in the base grid may create a situation in which some cells have a negative cell thickness. Although GRIDGEN does not fix this problem, it will indicate that negative cell thicknesses have been encountered. Users may wish to correct these errors or replace the top and bottom elevations for the layered quadtree grid with values calculated using some other form of interpolation. Replacement of cell top and bottom arrays (by simply replacing the array files) should be done prior to writing the cell connectivity (GRID_TO_USGDATA) and grid information so that the updated elevations are used to calculate connection properties.

Lastly, the QUADTREE_BUILDER block contains a record for GRID_DEFINITION_FILE. This record indicates the name of the definition file that will be created when the QUADTREE_BUILDER block is executed. For the example here, GRIDGEN will create a new definition file called quadtreegrid.dfn. This definition file will contain a new QUADTREE block, which references the new tree structure file. These files will be created when the following command is issued: “gridgen.exe quadtreebuilder quadtreebuilder.dfn”.

Descriptions of the records that compose the QUADTREE_BUILDER block are shown in table 4.

Table 4. List of records that compose the QUADTREE_BUILDER block.

Record name	Record type	Default	Record description
MODFLOW_GRID	<modflow_grid>	Required	MODFLOW_GRID block name
GRID_DEFINITION_FILE	<text>	Required	Name of the GRID_DEFINITION_FILE to create.
(As needed for any layer) REFINEMENT_FEATURES LAYER layer	List of <refinement_features>	Optional	List of REFINEMENT_FEATURES blocks to refine a layer.
(As needed for any layer) ACTIVE_DOMAIN LAYER layer	<active_domain>	Optional	ACTIVE_DOMAIN block to specify which cells in a layer are active.
SMOOTHING	<text>	None	Smoothing options are “none” or ”full”
HORIZONTAL_SMOOTHING_LEVEL	<positive integer>	1	Maximum level difference between two horizontally adjacent cells (normally 1).
VERTICAL_SMOOTHING_LEVEL	<positive integer >	1	Maximum level difference between two vertically adjacent cells (normally 1).
(For each layer) TOP LAYER layer	REPLICATE <modflow_grid> INTERPOLATE <modflow_grid> ASCIIGRID asciigridfile	Required	Apply one of three methods for assigning cell top elevations.
(For each layer) BOTTOM LAYER layer	REPLICATE <modflow_grid> INTERPOLATE <modflow_grid> ASCIIGRID asciigridfile	Required	Apply one of three methods for assigning cell bottom elevations.
AUTOALIGNMENT	<boolean>	False	If true, then align top *and* bottom elevations. If a larger cell overlies multiple smaller cells, then assign the larger cell bottom elevation to the tops of all underlying smaller cells. Likewise, if a larger cell underlies multiple smaller cells, then assign the top elevation of the larger cell to the bottom elevations of the smaller cells.

REFINEMENT_FEATURES

The REFINEMENT_FEATURES block is a mechanism for providing points, lines, and polygon features to the QUADTREE_BUILDER block, which then uses these features to refine the grid. Information about the refinement level for each feature set is also included in this block. The following are examples of REFINEMENT_FEATURES blocks that might be used with the QUADTREE_BUILDER block shown in the previous section:

```
BEGIN REFINEMENT_FEATURES ad_layer_1
  SHAPEFILE = shapefiles/active_domain_layer_1
  FEATURE_TYPE = polygon
  REFINEMENT_LEVEL = 1
END REFINEMENT_FEATURES

BEGIN REFINEMENT_FEATURES river
  SHAPEFILE = shapefiles/river
  FEATURE_TYPE = line
  REFINEMENT_LEVEL = 2
END REFINEMENT_FEATURES

BEGIN REFINEMENT_FEATURES wells_layer_1
  SHAPEFILE = shapefiles/wells_layer_1
  FEATURE_TYPE = point
  REFINEMENT_LEVEL = 2
END REFINEMENT_FEATURES

BEGIN REFINEMENT_FEATURES wells_layer_3
  SHAPEFILE = shapefiles/wells_layer_3
  FEATURE_TYPE = point
  REFINEMENT_LEVEL = 3
END REFINEMENT_FEATURES
```

Cells in the layered quadtree grid that contain the point, touch the line, or are within or touch the polygon will be refined to the specified REFINEMENT_LEVEL value. A description of the REFINEMENT_FEATURES block is shown in table 5.

Table 5. List of records that compose the REFINEMENT_FEATURES block.

Record name	Record type	Default	Record description
SHAPEFILE	<text>	Required	Name of shapefile
FEATURE_TYPE	<text>	Required	“point,” “line,” or “polygon”
REFINEMENT_LEVEL	<integer>	Required	Level of refinement to use
REFINE_LEVEL_BY_ATTRIBUTE	<text>	Optional	Attribute name in shapefile that contains the refinement level. The attribute must contain integer refinement values for each feature. If specified, this will override the REFINEMENT_LEVEL value.

ACTIVE_DOMAIN

When used with the QUADTREE_BUILDER block, the ACTIVE_DOMAIN block is used to determine which cells are active; these active cells are assigned a positive integer cell number. An active domain can be assigned for each layer. If no active domain is specified for a layer, then all cells in that layer are active by default. If an active domain is specified, then only the cells that intersect the ACTIVE_DOMAIN feature are active.

The following is an example of an ACTIVE_DOMAIN block:

```
BEGIN ACTIVE_DOMAIN active_domain_layer_1
  SHAPEFILE = shapefiles/active_domain_layer_1
  FEATURE_TYPE = polygon
  INCLUDE_BOUNDARY = True
END ACTIVE_DOMAIN

BEGIN ACTIVE_DOMAIN active_domain_layer_3
  SHAPEFILE = shapefiles/active_domain_layer_3
  FEATURE_TYPE = polygon
  INCLUDE_BOUNDARY = True
END ACTIVE_DOMAIN
```

The list of records that compose the ACTIVE_DOMAIN block is described in table 6.

Table 6. List of records that compose the ACTIVE_DOMAIN block.

Record name	Record type	Default	Record description
SHAPEFILE	<text>	Required	Name of shapefile
FEATURE_TYPE	<text>	Required	“point,” “line,” or “polygon”
INCLUDE_BOUNDARY	<boolean>	True	True or False. Determines whether cells partially intersected by a polygon boundary are included in the active domain.

GRID_TO_SHAPEFILE

The GRID_TO_SHAPEFILE block is an action block and can be used to create shapefiles for grids described with the MODFLOW_GRID and QUADTREE blocks. Note that all cells are written to the shapefile, even for multilayer models, which can result in polygons that overlap in map view. When loaded into a shapefile viewer, it may be necessary to select cell subsets by layer in order to make meaningful maps and plots.

The following example shows a GRID_TO_SHAPEFILE block for a MODFLOW_GRID with the name "basegrid." In this case, "basegrid" is imported from the basegrid.dfn definition file. The block specifies that the cells of the grid should be saved as polygons to the file named output_shapefiles/mfgrid:

```
LOAD basegrid.dfn

BEGIN GRID_TO_SHAPEFILE mfg-to-shapefile
  GRID = basegrid
  SHAPEFILE = output_shapefiles/mfg
  FEATURE_TYPE = polygon
END GRID_TO_SHAPEFILE
```

The following block shows another example that saves a layered quadtree grid to a set of points, each of which is located at the center of a grid cell:

```
LOAD quadtreegrid.dfn

BEGIN GRID_TO_SHAPEFILE quadtree2shapefile
  GRID = quadtreegrid
  SHAPEFILE = output_shapefiles/quadtreegrid
  FEATURE_TYPE = point
END GRID_TO_SHAPEFILE
```

The definitions for the list of records that compose the GRID_TO_SHAPEFILE block are listed in table 7.

Table 7. List of records that compose the GRID_TO_SHAPEFILE block.

Record name	Record value	Default	Record description
GRID	<text>	Required	Name of MODFLOW_GRID or QUADTREE block
SHAPEFILE	<text>	Required	Name of shapefile to create
FEATURE_TYPE	<text>	Required	Type of feature to write to the shapefile: "point," "line," or "polygon"

In the shapefiles created by this block, each feature (polygon, line, or point) contains several attributes, which are also written to the shapefile. For a MODFLOW_GRID, the attributes are

1. nodenumber: number of the cell;
2. layer: layer number of the cell;
3. row: row number of the cell;
4. col: column number of the cell;
5. child_location: this attribute is empty. It is included here to maintain consistency with shapefiles of QUADTREE grids;

6. top: top elevation of the cell;
7. bottom: bottom elevation of the cell;
8. delr: width of the cell in the x direction; and
9. delc: width of the cell in the y direction.

The attributes written for a QUADTREE grid are similar to those written for a MODFLOW_GRID. Only the leaves of the layered quadtree grid are stored in the shapefile. The main difference is that the created shapefile contains a child_location attribute, which is the list of quadrant sequence numbers. This string specifies the unique position of the quadtree cell within the base grid cell. It is also important to note that the values for the row and column refer to the row and column of the base grid cell. The attributes for a shapefile created from a QUADTREE grid are

1. nodenumber: number of the cell;
2. layer: layer number of the cell;
3. row: row number of the base grid cell;
4. col: column number of the base grid cell;
5. child_location: a string consisting of numbers from 1 to 4 defining the quadrants to locate the leaf node within the base grid cell;
6. top: top elevation of the cell;
7. bottom: bottom elevation of the cell;
8. delr: width of the cell in x-direction; and
9. delc: width of the cell in y-direction.

GRID_TO_USGDATA

The GRID_TO_USGDATA block is included specifically to create many of the discretization input arrays required by the MODFLOW–USG groundwater flow program (Panday and others, 2013). This block is an action block that writes ASCII text files containing information about the geometrical properties of cells and cell connections. An example of a GRID_TO_USGDATA block is shown below:

```
BEGIN GRID_TO_USGDATA qtree2usgdata
  GRID = quadtreegrid
  USG_DATA_PREFIX = output/usginput
  VERTICAL_PASS_THROUGH = true
END GRID_TO_USGDATA
```

Note that there are several records for the GRID_TO_USGDATA block (table 8). The GRID record is the name of the QUADTREE grid; the USG_DATA_PREFIX is the prefix name to use for the files that are created. The files created by GRIDGEN, when run with this block, are listed in table 9. Readers are referred to the MODFLOW–USG manual (Panday and others, 2013) for detailed descriptions of the arrays written to these files.

Table 8. List of records that compose the GRID_TO_USGDATA block.

Record name	Record value	Default	Record description
GRID	<text>	Required	Name of QUADTREE block
USG_DATA_PREFIX	<text>	Required	Filename prefix for files to create
VERTICAL_PASS_THROUGH	<boolean>	False	Option for connecting layers where intermediate layers are inactive.

Table 9. Files created by the GRID_TO_USGDATA block.

[NCELLS, number of active cells for the grid; CSR, compress sparse row; NCON, total number of connections between cells]

File	Array type (number of entries)	Array description
prefix.ia.dat	<int[NCELLS+1]>	CSR row pointer array
prefix.ja.dat	<int[NCON]>	CSR column index pointer array
prefix.iac.dat	<int[NCELLS]>	Number of entries for each row
prefix.area.dat	<real[NCELLS]>	Cell surface area in plan view
prefix.c1.dat	<real[NCON]>	Distance from cell center to shared face of connected cell
prefix.c2.dat	<real[NCON]>	Distance from center of connected cell to shared face
prefix.fahl.dat	<real[NCON]>	Cross sectional area of the connection. For horizontal connections, this is calculated based on the average thickness of the connected cells.
prefix.fldr.dat	<int[NCON]>	Direction indicator for each connection. 0 is diagonal, 1, 2, and 3 are for the positive *x*, *y*, and *z* directions. –1, –2, and –3 are for the negative *x*, *y*, and *z* directions.
prefix.gnc.dat		List of ghost-node correction information. Note that the ghost-node correction information is only calculated for horizontal connections. Ghost-node correction information is not calculated for vertical cell connections. QUADTREE grids should be smoothed to a level difference of one in the horizontal direction for the information in this file to be useful.
prefix.nodesperlay.dat	<int[NLAY]>	Number of cells for each layer
prefix.nod		The first line in this file is the number of active cells and the number of connections. This file then contains one line for each active cell. Data for each line include: (1) cell number, (2) layer number, (3) centroid x, (4) centroid y, (5) centroid z, (6) delta x spacing, (7) delta y spacing, and (8) delta z spacing.

The VERTICAL_PASS_THROUGH record indicates how vertical connections will be handled for cells that are outside of the active domain. In some cases, the flexibility to specify the active domain for each model layer introduces complexity in determining vertical cell connections. For example, there may be a situation in which model layers 1 and 3 represent aquifers and both layers are active everywhere in the base grid. Model layer 2, however, represents a discontinuous confining unit that is only present in some areas. If the VERTICAL_PASS_THROUGH option is set to “true,” then GRIDGEN will automatically connect model layers 1 and 3 within the inactive area of model layer 2. If the VERTICAL_PASS_THROUGH option is set to “false,” then there will be no direct connections between model layers 1 and 3.

GRID_TO_VTKFILE

The GRID_TO_VTKFILE block is an action block that can be used to create a Visualization Toolkit (VTK) file for the specified grid. An example of a GRID_TO_VTK block is shown below:

```
BEGIN GRID_TO_VTKFILE grid02qtg-to-vtkfile
  GRID = grid02qtg
  VTKFILE = output_vtkfiles/grid02qtg
  SHARE_VERTEX = false
END GRID_TO_VTKFILE
```

The GRID_TO_VTKFILE block contains two records. Descriptions for these records are shown in table 10. This block can be used with a MODFLOW_GRID or a QUADTREE grid as input. The output file created by this action block can be used as input for visualization software.

Table 10. List of records that compose the GRID_TO_VTKFILE block.

Record name	Record value	Default	Record description
GRID	<text>	Required	Name of MODFLOW_GRID or QUADTREE block
VTKFILE	<text>	Required	Filename prefix for VTK file to create
SHARE_VERTEX	<boolean>	False	If true, then the elevation for each cell vertex will be interpolated and vertices will be shared among neighboring cells. This results in a smooth surface and a smaller VTK file. If not, each cell in the VTK file will be given 8 vertices resulting in a stair-stepped cell representation.

GRID_INTERSECTION

A common task in the development of groundwater models is determining the intersection properties of hydrologic features with the model grid. To accomplish this task with GRIDGEN, features within shapefiles can be intersected with a grid by specifying a GRID_INTERSECTION block. All of the intersection routines are two dimensional; there is no use of elevations within the routines to determine intersection properties. Instead, a layer number must be provided for each intersection. The GRID_INTERSECTION block is an action block and can therefore be executed by the GRIDGEN program. Three types of features can be intersected with a grid: points, lines, and polygons. Results from the intersection are written to a shapefile and to an ASCII text file. Table 11 lists the records of the GRID_INTERSECTION block. The remainder of this section presents details of the three different types of intersections: point-grid, line-grid, and polygon-grid.

Table 11. List of records that compose the GRID_INTERSECTION block.

Record name	Record value	Default	Record description
GRID	<text>	Required	Name of grid block
LAYER	<integer>	Required	Layer of the grid to intersect with the features
SHAPEFILE	<text>	Required	Name of grid shapefile containing the features to intersect with the grid
FEATURE_TYPE	<text>	Required	Type features to intersect with the grid: “point,” “line,” or “polygon”
OUTPUT_FILE	<text>	Required	Name of the ASCII output file to create
OUTPUT_SHAPEFILE	<text>	Optional	Name of the shapefile to create
ATTRIBUTES	<text>	“All”	List of attribute names from the input shapefile to include in the output shapefile created by the intersection. Note that attribute names are case sensitive and that the terms “all” or “none” can also be entered here.

Point-Grid Intersection

When point features are intersected with a grid, the intersection routine determines the cell number that contains each point. The results from the intersection are then written to an ASCII text file and to a new shapefile. An example of a point-grid intersection block is provided below. In this example, the first layer of the quadtree grid is intersected with a shapefile containing a list of points.

```
BEGIN GRID_INTERSECTION well1_intersect
  GRID = quadtreegrid
  LAYER = 1
  SHAPEFILE = shapefiles/wells_layer_1
  FEATURE_TYPE = point
  OUTPUT_FILE = output_intersection/well1_intersect.ifo
  OUTPUT_SHAPEFILE = output_intersection/well1_intersect
END GRID_INTERSECTION
```

The resulting ASCII file will contain at least two columns of information, the cell number and the point identifier. Other attribute information from the original point shapefile would also be included if requested. In this case, a point shapefile would also be created and would have the same attributes as those listed in the ASCII text file.

There are special cases for a point-grid intersection that involve the point coinciding with a cell edge or corner. For these special cases in which a point falls on the shared border of two cells, GRIDGEN returns the cell with the smaller cell number.

Line-Grid Intersection

Intersection of lines with a grid provides information about the cells that touch the lines, and information about the line lengths within each cell is also calculated and provided by GRIDGEN. An example of a line-grid intersection block is provided below. In this example, the first layer of the quadtree grid is intersected with a shapefile containing a set of lines from the shapefile contained in shapefiles/river. As for the point-grid intersection, the output is saved to both an ASCII file and a shapefile, except that the shapefile will contain lines instead of points. Both file formats are discussed below.

```
BEGIN GRID_INTERSECTION river_intersect
  GRID = quadtreegrid
  LAYER = 1
  SHAPEFILE = shapefiles/river
  FEATURE_TYPE = line
  OUTPUT_FILE = output_intersection/river_intersect.ifo
  OUTPUT_SHAPEFILE = output_intersection/river_intersect.shp
END GRID_INTERSECTION
```

The resulting ASCII text file from this intersection will contain multiple columns of information, including the cell number, line identifier, length of the line in the cell, starting and ending distance along the line, and other requested attributes from the line shapefile. Note that a single line feature may intersect a cell multiple times. If this is the case, then each intersection feature will be reported as a record. These records will share the same cell number and line identifier but will differ in their starting and ending distances. The shapefile created from the line-grid intersection contains the same information as the ASCII output.

There are several special cases for the line-grid intersection. The location of a line within the grid is determined by the locations of vertices composing the line. When a line segment coincides with the shared border of two cells, then this line segment will be assigned to the cell having the smaller cell number. If a line has two parts, with one part inside the cell and the other part on the edge of that cell, then GRIDGEN will either (1) split the line into two lines if the adjacent cell has a smaller cell number, or (2) leave the line undivided if the adjacent cell does not have a smaller cell number.

Polygon-Grid Intersection

Intersection of polygons with a grid provides information about the cells that are within or touching the polygons, and information about the polygon areas within each cell is also calculated and provided by GRIDGEN. An example of polygon-grid intersection block is provided below. In this example, the first layer of the quadtree grid is intersected with a shapefile containing multiple polygons. Intersection output is saved to both an ASCII file and a polygon shapefile.

```
BEGIN GRID_INTERSECTION recharge_intersect
  GRID = quadtreegrid
  LAYER = 1
  SHAPEFILE = shapefiles/recharge
  FEATURE_TYPE = polygon
  OUTPUT_FILE = output_intersection/recharge_intersect.ifo
  ATTRIBUTES = rech_zone
  OUTPUT_SHAPEFILE = output_intersection/recharge_intersect.shp
END GRID_INTERSECTION
```

The resulting ASCII text file from this intersection will contain multiple columns of information, including the cell number, polygon identifier, area of the polygon within the cell, and requested attributes. The shapefile representation of the intersection will contain the same information.

There is one special case for the polygon intersection when a polygon touches only the edge of a cell. In this case, GRIDGEN will not save an entry for that polygon, because the intersection area is zero.

Example

In this example, a layered quadtree grid is generated using spatial information for the Biscayne aquifer in southern Florida. The single layer quadtree grid designed here is similar to the grid used for quadtree groundwater flow simulation described in the MODFLOW–USG documentation (Panday and others, 2013). The grids are not identical, however, as the intersection and cell numbering routines in GRIDGEN are different from the ones used to create the quadtree grid described in Panday and others (2013).

This example consists of four steps, each involving the execution of a GRIDGEN action block. In the first step, a QUADTREE_BUILDER block is used to create a layered quadtree grid. Next, the GRID_TO_USGDATA is used to write information about the grid to a set of files. In the third step, point and polygon shapefiles are created for the layered quadtree grid. Lastly, the layered quadtree grid is intersected with point, line, and polygon shapefiles.

Generating the Layered Quadtree Grid

The QUADTREE_BUILDER block is used with GRIDGEN to create the layered quadtree grid. The base MODFLOW_GRID consists of 1 layer, 100 rows, and 58 columns (fig. 2*A*). Each cell in the base grid is 800 meters (m) on a side. The spatial features used with the QUADTREE_BUILDER block include an active domain, lines representing the freshwater canals, lines representing the tidal canals and the coastline, and polygons surrounding municipal wells (fig. 2*A*). Points could also have been used as refinement features for the municipal wells, but use of polygons provides more control over the size of the refinement area near wells. These features are used with a REFINEMENT_LEVEL of 4 to create cells in the layered quadtree grid with a minimum width of 50 m (with a base grid cell size of 800 m, a refinement level of 4 is equal to 50 m). The resulting layered quadtree grid is shown in figure 2*B*.

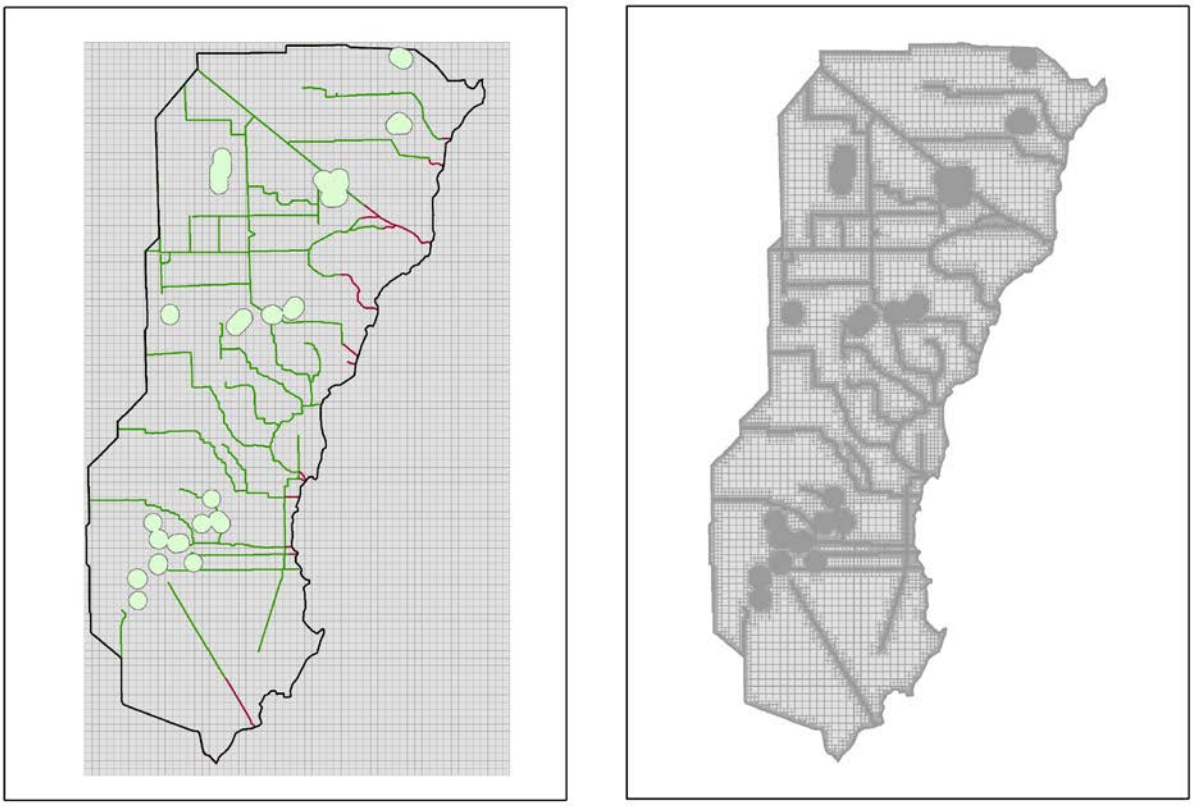


Figure 2. Diagram showing *A*, features and base grid (MODFLOW_GRID) used with the QUADTREE_BUILDER block in GRIDGEN to create *B*, the layered quadtree grid. Note that quadtree cells outside of the active domain are given a cell number of –1. Although these cells are part of the quadtree grid and included in the tree structure file, they would not be included as part of the flow simulation.

The definition file (named "action01_buildqtg.dfn") used to create the layered quadtree grid shown in figure 2*B* is as follows:

```
#the "#" symbol indicates the line is a comment

BEGIN MODFLOW_GRID grid01mfg
  ROTATION_ANGLE = 0.
  X_OFFSET = 543750
  Y_OFFSET = 2792250
  LENGTH_UNIT = undefined
  NLAY = 1
  NROW = 100
  NCOL = 58
  DELR = CONSTANT 800.
  DELC = CONSTANT 800.
  TOP = OPEN/CLOSE grid01mfg.top.dat
  BOTTOM LAYER 1 = OPEN/CLOSE grid01mfg.bot.dat
END MODFLOW_GRID

BEGIN QUADTREE_BUILDER buildqtg
  MODFLOW_GRID = grid01mfg
  ACTIVE_DOMAIN LAYER 1 = my_active_domain
```

```
  REFINEMENT_FEATURES LAYER 1 = MD_Canals chd_line well_buffer
  SMOOTHING = full
  SMOOTHING_LEVEL_HORIZONTAL = 1
  SMOOTHING_LEVEL_VERTICAL = 1
  GRID_DEFINITION_FILE = grid02qtg.dfn
  TOP LAYER 1= REPLICATE grid01mfg
  BOTTOM LAYER 1 = REPLICATE grid01mfg
END QUADTREE_BUILDER

BEGIN REFINEMENT_FEATURES well_buffer
  SHAPEFILE = shapefiles/wells_1000m_buffer_Multipart
  FEATURE_TYPE = polygon
  REFINEMENT_LEVEL = 4
END REFINEMENT_FEATURES

BEGIN REFINEMENT_FEATURES chd_line
  SHAPEFILE = shapefiles/chd_line
  FEATURE_TYPE = line
  REFINEMENT_LEVEL = 4
END REFINEMENT_FEATURES

BEGIN REFINEMENT_FEATURES MD_Canals
  SHAPEFILE = shapefiles/MD_Canals_50m_v2
  FEATURE_TYPE = line
  REFINEMENT_LEVEL = 4
END REFINEMENT_FEATURES

BEGIN ACTIVE_DOMAIN my_active_domain
  SHAPEFILE = shapefiles/active_domain
  FEATURE_TYPE = polygon
  INCLUDE_BOUNDARY = True
END ACTIVE_DOMAIN
```

The layered quadtree grid is generated by GRIDGEN with the command: “gridgen.exe buildqtg action01_buildqtg.dfn”. When this command is executed from a command line, GRIDGEN writes information to indicate that a layered quadtree grid was created and written to the file “grid02qtg.dfn”. Supporting information is also written to a tree structure file (“grid02qtg.tsf”), an array file containing the top elevation for each cell (“grid02qtg.top1.dat”), and an array file containing the bottom elevation for each active cell (“grid02qtg.bot1.dat”). Note that the top and bottom cell elevations are replicated (not interpolated) from the top and bottom elevations from the base grid.

Smoothing is an important part of the grid generation process. This is controlled by the SMOOTHING and SMOOTHING_LEVEL_HORIZONTAL keywords in the QUADTREE_BUILDER block. A SMOOTHING_LEVEL_VERTICAL record is not needed for this example because the grid contains only one layer. Smoothing can be deactivated by changing the SMOOTHING value from “full” to “none,” but for most cases, smoothing levels of 1 should be used.

Writing Grid Information

Cell connectivity and other information about the grid are written to ASCII files using the GRID_TO_USGDATA block. The definition file (“action02_writeusgdata.dfn”) for this action contains the following lines:

```
LOAD grid02qtg.dfn
BEGIN GRID_TO_USGDATA grid02qtg-to-usgdata
  GRID = grid02qtg
  USG_DATA_PREFIX = output_usgdata/grid02qtg
END GRID_TO_USGDATA
```

The information is written by GRIDGEN when the command “gridgen.exe grid02qtg-to-usgdata action02_writeusgdata.dfn” is executed. The result of this action block is the creation of 12 files within the “output_usgdata” folder. The information contained in these files is described in table 9.

Some of the information created by the GRID_TO_USGDATA action block requires cell top and bottom elevations. In this example, cell tops and bottoms are contained in “grid02qtg.top1.dat” and “grid02qtg.bot1.dat”. In the present GRIDGEN version, the QUADTREE_BUILDER block replicates cell top and bottom elevations from the base MODFLOW grid. In many instances, users may prefer to interpolate cell top and bottom elevations outside of GRIDGEN using alternative methods. If those interpolated cell top and bottom elevations can be converted into array files, then the top and bottom array files created by the QUADTREE_BUILDER block (“grid02qtg.top1.dat” and “grid02qtg.bot1.dat”) should be replaced by user-provided array files with the same name. If this is done prior to running the GRID_TO_USGDATA block, then cell connectivity and grid properties will be calculated using the interpolated cell top and bottom elevations.

Creating Shapefiles of the Grid

A point and polygon shapefile can be created for the layered quadtree grid using the GRID_TO_SHAPEFILE block. The definition file (“action03_shapefile.dfn”) for these actions contains the following lines:

```
LOAD grid02qtg.dfn

BEGIN GRID_TO_SHAPEFILE grid02qtg-to-pointshapefile
  GRID = grid02qtg
  SHAPEFILE = output_shapefiles/grid02qtg_pts
  FEATURE_TYPE = point
END grid_to_shapefile

BEGIN GRID_TO_SHAPEFILE grid02qtg-to-polyshapefile
  GRID = grid02qtg
  SHAPEFILE = output_shapefiles/grid02qtg
  FEATURE_TYPE = polygon
END grid_to_shapefile
```

The shapefiles are written by GRIDGEN to the output_shapefiles folder using the following commands: “gridgen.exe grid02qtg-to-pointshapefile action03_shapefile.dfn” and “gridgen.exe grid02qtg-to-polyshapefile action03_shapefile.dfn”.

Intersecting the Grid

The last part of this example involves intersection of the layered quadtree grid with spatial features contained in shapefiles. The definition file for intersecting the grid with several different shapefiles (“action04_intersect.dfn”) contains the following lines:

```
LOAD grid02qtg.dfn

BEGIN GRID_INTERSECTION canal_grid02qtg_lay1_intersect
  GRID = grid02qtg
  LAYER = 1
  SHAPEFILE = shapefiles/MD_Canals_50m_v2
  FEATURE_TYPE = line
  OUTPUT_FILE = output_intersection/canal_grid02qtg_lay1_intersect.ifo
  ATTRIBUTES = HYDR_COND BOT_ELEV TOP_WIDTH DB_NAME
  OUTPUT_SHAPEFILE = output_intersection/canal_grid02qtg_lay1_intersect.shp
END GRID_INTERSECTION

BEGIN GRID_INTERSECTION well_grid02qtg_lay1_intersect
  GRID = grid02qtg
  LAYER = 1
  SHAPEFILE = shapefiles/county_wells_NAD_27_filtered
  FEATURE_TYPE = point
  OUTPUT_FILE = output_intersection/well_grid02qtg_lay1_intersect.ifo
  ATTRIBUTES = WellName
  OUTPUT_SHAPEFILE = output_intersection/well_grid02qtg_lay1_intersect.shp
END GRID_INTERSECTION

BEGIN GRID_INTERSECTION poly_grid02qtg_lay1_intersect
  GRID = grid02qtg
  LAYER = 1
  SHAPEFILE = shapefiles/twopolygons
  FEATURE_TYPE = polygon
  OUTPUT_FILE = output_intersection/poly_grid02qtg_lay1_intersect.ifo
  ATTRIBUTES = FID
  OUTPUT_SHAPEFILE = output_intersection/poly_grid02qtg_lay1_intersect.shp
END GRID_INTERSECTION

BEGIN GRID_INTERSECTION chd_grid02qtg_lay1_intersect
  GRID = grid02qtg
  LAYER = 1
  SHAPEFILE = shapefiles/chd_line
  FEATURE_TYPE = line
  OUTPUT_FILE = output_intersection/chd_grid02qtg_lay1_intersect.ifo
  #attributes = FID
  OUTPUT_SHAPEFILE = output_intersection/chd_grid02qtg_lay1_intersect.shp
END GRID_INTERSECTION

BEGIN GRID_INTERSECTION lu2008_grid02qtg_lay1_intersect
  GRID = grid02qtg
  LAYER = 1
  SHAPEFILE = shapefiles/SFWMD_2008_09_LCLU_2004FILLED_MISSING
  FEATURE_TYPE = polygon
  OUTPUT_FILE = output_intersection/lu2008_grid02qtg_lay1_intersect.ifo
  ATTRIBUTES = FID
  OUTPUT_SHAPEFILE = output_intersection/lu2008_grid02qtg_lay1_intersect.shp
END GRID_INTERSECTION
```

Each one of the GRID_INTERSECTION blocks requires a separate execution of the GRIDGEN program. To execute the first GRID_INTERSECTION block (called canal_grid02qtg_lay1_intersect), the following command would be executed: “gridgen.exe canal_grid02qtg_lay1_intersect

action04_intersect.dfn". The rest of the intersection blocks would be executed following similar commands using the individual names of the GRID_INTERSECTION blocks.

Summary and Conclusions

This report describes the GRIDGEN computer program, which can be used to construct unstructured finite-volume model grids. This first GRIDGEN version (Version 1.0) can be used to construct layered quadtree grids. Once a layered quadtree grid is constructed, GRIDGEN can write cell connectivity and other discretization information for the grid to ASCII files in a format required by the MODFLOW–USG groundwater flow simulation program. GRIDGEN can also create shapefiles of the grid and intersect the grid with other shapefiles that may contain hydrologic features, such as streams, rivers, or landuse polygons, for example. The GRIDGEN program is intended to serve as one part of a set of user tools for creating input files for MODFLOW–USG and other simulation programs.

References Cited

Environmental Systems Research Institute, Inc. (ESRI), 1998, ESRI Shapefile technical description: Accessed December 18, 2013, at *http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf*.

Environmental Systems Research Institute, Inc. (ESRI), 2014, ESRI ASCII raster format: Accessed March 25, 2014, at *http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/ESRI_ASCII_raster_format/009t0000000z000000/*.

Harbaugh, A.W., 2005, MODFLOW–2005, the U.S. Geological Survey modular ground-water model—The Ground-Water Flow Process: U.S. Geological Survey Techniques and Methods, book 6, chap. A16, variously paged.

Panday, Sorab, Langevin, C.D., Niswonger, R.G., Ibaraki, Motomu, and Hughes, J.D., 2013, MODFLOW–USG version 1: An unstructured grid version of MODFLOW for simulating groundwater flow and tightly coupled processes using a control volume finite-difference formulation: U.S. Geological Survey Techniques and Methods, book 6, chap. A45, 66 p.

Warmerdam, F., 1998, Shapefile C Library, *http://shapelib.maptools.org*.

Publishing support provided by the U.S. Geological Survey
Science Publishing Network, Raleigh and Reston
Publishing Service Centers

For more information about this report, contact:
Office of Groundwater
U.S. Geological Survey
411 National Center
12201 Sunrise Valley Drive
Reston, VA 20192
(703) 648-5001
http://water.usgs.gov/ogw/contact.html

ISSN 2331-1258 (online)
http://dx.doi.org/10.3133/ofr20141109