

**USER'S MANUAL OF THE
MESHMAKER v2.0 CODE:
A MESH GENERATOR FOR DOMAIN
DISCRETIZATION IN SIMULATIONS
OF THE TOUGH+ AND TOUGH2/3
FAMILIES OF CODES**

George J. Moridis

*Energy Geosciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720*

January 2024

This work was supported by the Assistant Secretary for Fossil Energy, Office of Natural Gas and Petroleum Technology, through the National Energy Technology Laboratory, under the U.S. Department of Energy, Contract No DE-AC02-05CH11231.

PAGE LEFT INTENTIONALLY BLANK

User's Manual of the MESHMAKER V2.0 Code: A Mesh Generator For Domain Discretization In Simulations of the TOUGH+ and TOUGH2/3 Families Of Codes

George J. Moridis

*Earth Sciences Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, California*

Abstract

MESHMAKER V2.0 is a code that describes the system geometry and discretizes the domain in problems of flow and transport through porous and fractured media that are simulated using the TOUGH+ [Moridis and Pruess, 2014], TOUGH2 [Pruess *et al.*, 1999; 2012] or TOUGH3 [Jung *et al.*, 2018] families of codes. It is a significantly modified and drastically enhanced version of an earlier simpler facility that was embedded in the TOUGH2 codes [Pruess *et al.*, 1999; 2012], from which it could not be separated. MESHMAKER V2.0 is an upgrade from the MESHMAKER V1.5 [Moridis, 2016], and the improvements focus on the simplification of the data inputs and on the enhancement of gridding options and capabilities in fractured media. The code (**MeshMakerV2.f90**) is a stand-alone product written in FORTRAN 95/2003, and can be run on any computational platform (workstations, PC, Macintosh). Its architecture follows the tenets of Object-Oriented Programming, has a modular structure and can perform a number of mesh generation and processing operations, including the creation of heterogeneous subdomains and boundaries within the simulated domains. It can generate two-dimensional radially symmetric (r,z) meshes, and one, two-, and three-dimensional rectilinear (Cartesian) grids in (x,y,z). The code generates the file **MESH**, which includes all the elements and connections that describe the discretized simulation domain and conforming to the requirements of the TOUGH+ and TOUGH2/3 codes. Multiple-porosity processing for simulation of flow in naturally fractured reservoirs can be invoked by means of a keyword MINC, which stands for *Multiple INteracting Continua*. The MINC process operates on the data of the primary (porous medium) mesh as provided on disk file **MESH**, and generates a secondary mesh containing fracture and matrix elements with identical data formats on file **MINC**.

PAGE LEFT INTENTIONALLY BLANK

Table of Contents

Abstract.....	iii
LIST OF FIGURES	vii
1.0. Introduction.....	1
1.1. Background	1
1.2. The MESHMAKER V2.0 Application	2
1.3. Compiling and Running the MESHMAKER V2.0 Application.....	3
1.4. Improvements in the MESHMAKER V2.0 and Differences from MESHMAKER V1.5	3
1.5. MESHMAKER V2.0 User's Manual: Conventions and General Comments	4
2.0. Code Description and Basic Inputs in MESHMAKER V2.0	6
2.1. TOUGH+ and TOUGH2/3 Conventions for Processing Discretization Data.....	6
2.2. Basic Inputs for the MeshMakerV2 . f90 Code.....	7
3.0. Description of Heterogeneity in MESHMAKER V2.0	10
3.1. Inputs for the Definition of Heterogeneous Regions	10
3.2. Inputs for the Definition of Boundaries	14
3.3. Inputs for the Definition of Excluded Regions	16
4.0. Domain Discretization and Grid Generation in MESHMAKER V2.0	20
4.1. Inputs for Cylindrical Grid Construction.....	21
4.2. Inputs for Cartesian (Rectilinear) Grid Construction.....	29
4.2. Inputs for MINC Processing for Fractured Media.....	34
Acknowledgements	40
References.....	42

PAGE LEFT INTENTIONALLY BLANK

List of Figures

Figure 1.	Test MM_R1: examples of simple MMV2 input files describing a regular cylindrical domain without heterogeneities, i.e. no regions with different names, no boundaries and no exclusions. Test MM_R1a has equally-sized subdivisions of the z-dimension; the z-dimension in Test MM_R1b is discretized in unequally-sized increments. The discretization of the radial dimension includes all the possible options available in MMV2. The domains in both cases are discretized by layers. Note that all entries beginning with the '!' characters and beyond are comments for the user's convenience, are not read by the code and can be omitted.	25
Figure 2.	Test MM_R2: MMV2 input file for the discretization a heterogeneous cylindrical domain comprising two different regions (media) medium and including no boundaries. The discretization of the radial dimension includes all the possible options available in MMV2. The domain is discretized by columns.	26
Figure 3.	Test MM_R3: MMV2 input file for the discretization a heterogeneous cylindrical domain comprising six different regions (media) and including two boundaries. The discretization of the vertical dimension includes three different <code>LAYER</code> data sets. The domain is discretized by columns.	27
Figure 5.	Test MM_C1: Examples of simple MMV2 input files describing a regular Cartesian domain without heterogeneities, i.e., no regions with different names, no boundaries and no exclusions. Test MM_C1a has unequally-sized subdivisions of the z-dimension; the z-dimension in Test MM_C1b is discretized in equally-sized increments. All entries beginning with the '!' characters and beyond are comments for the user's convenience, are not read by the code and can be omitted.	31
Figure 6.	Test MM_C2: MMV2 input file for a regular Cartesian domain described by a single medium and including a rectangular boundary.	31
Figure 7.	Test MM_C3 (a variation of Test MM_C2): the MMV2 input file for irregularly-shaped Cartesian domains that involves a single medium, includes a rectangular boundary and its shape is formed by a rectangular exclusion.	32
Figure 8.	Test MM_C4: MMV2 input file for a Cartesian representation of a cylindrical domain with a horizontal main axis. The resulting Cartesian domain can be used for the simulation of a horizontal cylindrical homogeneous core and its holding container (defined the cylindrical boundary 'OutBB') when gravity cannot be ignored and its shape is formed by a cylindrical exclusion.	32
Figure 9.	Test MM_C5: An example of a MMV2 input file for the creation of a large Cartesian 3D grid with six heterogeneous regions and two different boundaries.	33
Figure 10.	Test MM_FC1: the MMV2 single input file for (a) the discretization a heterogeneous Cartesian domain comprising three different regions (of which one is fractured) and a fractured boundary and (b) the subsequent creation of the corresponding grid for the fractured domain. The primary (unfractured) grid is stored in the file MESH , and the fractured grid is stored in the file MINC . For the development of the fractured grid, the following are needed: (a) <code>media_by_number = .TRUE.</code> , (b) <code>FracPreparation = .TRUE.</code> , (c) the names of the fractured subdomains must begin with a 'F' or 'f' (all three denoted by red) and (d) the MINC data block (in blue) must be present immediately after the discretization specification data block <code>XYZ</code>	37
Figure 11.	Test MM_FC2: the MMV2 two separate input files needed for (a) the discretization a heterogeneous Cartesian domain comprising six different regions (of which only one is fractured) and an unfractured boundary and (b) the subsequent creation of the the corresponding grid for the fractured domain. The primary (unfractured) grid is obtaining by using the first (top) input file and is stored in the file MESH ; the fractured grid is from	

the second (lower, in blue and red colors) input file and is stored in the file **MINC**. For the development of the fractured grid, the following are needed in the top input file:
(a) `media_by_number = .TRUE.`, (b) `FracPreparation = .TRUE.` and (c) the names of the fractured subdomains must begin with a 'F' or 'f' (all three denoted by red).
For the second run, the first two lines (in blue) of the input file must be the same as in the first input file. The same results can be obtained from a single input file if the **MINC** data block (in red) is placed immediately after the end of the **XYZ** discretization datablock in the first input file (see Figure 10)..... 38

1.0. Introduction

1.1. Background

The TOUGH family of codes [Pruess *et al.*, 1999; 2012] and its most recent TOUGH+ [Moridis and Pruess, 2014] successor generation of codes were developed for the simulation of multi-component, multiphase fluid and heat flow developed at the Lawrence Berkeley National Laboratory (LBNL). These codes are based on the Integral Finite Difference space discretization method [Narasimhan and Witherspoon, 1976], a finite-volume method. The corresponding meshes that are needed for such simulations are quite different from the ones created by standard gridding software because they do not use global coordinates, but only local coordinates that describe the relative positions of the centers of the gridblock elements. This precludes the use of common gridding applications and necessitates specialized software.

Pruess *et al.* [1999; 2012] provided a simple mesh-making facility named MESHMAKER (hereafter referred to as MM1) for use with the TOUGH family of codes. The MM1 facility is not a stand-alone code, but it is completely integrated within the TOUGH2 code [Pruess *et al.*, 1999; 2012]. While useful in creating TOUGH meshes, MM1 has some serious shortcomings. It is applicable to homogeneous domains involving a single medium (rock type) and is limited to 5-character element names: its numbering system is inflexible (proceeding invariably along the x , y and z directions, always in that order), thus often preventing the computational and memory savings that could be attained during the matrix solving process by a reduction in the matrix bandwidth (a direct result of a more efficient element numbering). Additionally, the MM1 grids cannot provide the necessary information needed for the description of the more complex boundary specifications that are available options in the TOUGH+ family of codes [Moridis *et al.*, 2008; Moridis and Pruess, 2014]. For the solution of problems involving heterogeneous systems of complex

geometry, other gridding software—e.g., WinGridder [Pan, 2008]—have to be used, or the MM1-created grids have to be modified either manually or by using mission-specific software.

The earlier MESHMAKER V1.5 application [Moridis, 2016] was developed to address the shortcomings of MM1, and represented a significant advancement in domain discretization and gridding of domains used in TOUGH+ and TOUGH2 simulations. Unlike MM1, this was a stand-alone piece of software that operated independently of the these codes and allowed (a) the creation of grids comprising significantly larger numbers of elements than those possible with MM1 and (b) the easy description of heterogeneity by the assignment of an arbitrary number of subdomains of complex geometrical shapes representing domain regions and boundaries (constant and time-variable).

1.2. The MESHMAKER V2.0 Application

MESHMAKER V2.0 (hereafter referred to as MMV2) is a significantly enhanced version of the earlier MESHMAKER V1.5 application [Moridis, 2016], incorporates new capabilities in addition to expanding all the capabilities of the earlier version, and was developed for the discretization of domains involved in simulations of all members of the TOUGH family of codes.

As was the case in the earlier MESHMAKER V1.5 [Moridis, 2016], MMV2 can describe large (in terms of number of elements, theoretically in excess of 10^9 elements) systems with heterogeneous subdomains of arbitrary geometry and complex boundary conditions. It is written in standard FORTRAN 95/2003, and can be run on any computational platform (workstations, PC, Macintosh). It has a modular architecture that follows the tenets of Object-Oriented programming. It can create very large grids consisted of elements with 5- or 8-character-long names, with any number of heterogeneous subdomains of complex geometries, and with any number of time-invariant or transient boundaries.

MMV2 generates grids that are directly usable by all members of the TOUGH+ and of the conventional TOUGH2 and TOUGH3 families of codes for no-flow (Neuman-type) boundary systems. The complex boundary subdomains that it can create (usually involving constant conditions, i.e., of the Dirichlet type) are seamlessly used by the TOUGH+ codes, but *may in some cases* require some editing and manipulation for TOUGH2/3 simulations. The element numbering in the meshes created by MMV2 begins on the axis with the smallest number of subdivisions and progresses along axes with increasing numbers of subdivisions, thus ensuring minimum bandwidth of the matrices to be solved in TOUGH+/TOUGH2/3 simulations and significant computational and memory efficiencies. It permits up to 1000 elements with the same non-numerical part of their names (see Moridis and Pruess [2014]) at the same elevations, thus significantly simplifying initialization and gravity equilibration operations in large 2D and 3D systems.

MeshMakerV2.f90, the FORTRAN90/2003 code for the MMV2 application, is distributed as part of the TOUGH+ v1.5 [Moridis and Pruess, 2014] core code, the User's Manual of which incorporates a complete documentation of the gridding facility. Because of the potential usefulness of MMV2 application to users of the standard TOUGH2/3

families of codes, **MeshMakerV2.f90** is also distributed by LBNL as an independent separate entity, hence the need for this User's Manual that includes a full description of the code, its input requirements and capabilities, as well as examples of input files. Appropriate examples with sample input files (and the corresponding outputs) are also included in the electronic directory that is distributed with the code.

1.3. Compiling and Running the MESHMAKER V2.0 Application

As discussed already, the software of the distributed MMV2 application comprises a single file, the **MeshMakerV2.f90** code (i.e., an autonomous code unit), written in standard FORTRAN 95/2003. It has been designed for maximum portability, and runs on any computational form (Unix and Linux workstations, PC, Macintosh) for which such compilers are available. Compiling and running the code involves the standard procedures that are dictated by the requirements of the specific compiler used by the user.

NOTE: In compiling **MeshMakerV2.f90**, *it is important that the free-format source code option be invoked for proper compilation of the FORTRAN 95/2003 code.*

1.4. Improvements in the MESHMAKER V2.0 and Differences from MESHMAKER V1.5

While the internal processing of data in MMV2 is largely the same as in the earlier MESHMAKER V1.5 [Moridis, 2016] version, there are important enhancements and modifications that aim to improve the data input process, streamline the code structure, optimize the data processing and increase the execution speed. Additionally, MMV2 includes substantial new capabilities for the creation of complex heterogeneous grids of intricate shapes and for the description of fractured media. The main new changes and features are the following:

- (1) The fixed formats that were used for data inputs in the older MM1 and in MESHMAKER V1.5 versions – which were often the cause of mistakes and frustration when the fixed format fields were accidentally misaligned – have been replaced by free formats. Fixed formats are used *only* for reading the keywords that identify the various input data blocks.
- (2) Heterogeneous subdomains and boundaries may have rectangular, cylindrical (with various orientation options) or spherical shapes when Cartesian coordinates are involved, and cylindrical and spherical shapes for cylindrical coordinates.
- (3) Furthermore, the capability to describe *excluded regions* (and removing them from the elements and the connections of the grid) was added to MMV2, allowing the description of complex domains of irregular shapes.

- (4) Significant new grid capabilities for the description of fractured media were added. MMV2 can create MINC grids that can involve any number of fractured and unfractured regions (subdomains and boundaries) in any part of the domain and provides multiple options to describe the fracture-matrix communication and the interactions of the heterogeneous subdomains.
- (5) MMV2 removes all connections between inactive elements, leading to lower memory requirements for domains with substantial boundary surfaces.
- (6) For enhanced clarity, the beginning and the end of the various input data blocks in MMV2 are clearly identified.
- (7) To prevent errors that can result in erroneous grids, a large number of internal checks have been added that ensure the validity of the data inputs and cause the interruption of the process (and the printing of clarifying error messages) if mistakes in the input data are detected.
- (8) A large part of the **MeshMakerV2.f90** application was rewritten using the most advanced features of FORTRAN 95/2003 to replace older and sometimes obsolete code (e.g., replacing common blocks by modules), thus significantly enhancing the computational efficiency of the code.

1.5. MESHMAKER V2.0 User's Manual: Conventions and General Comments

The following conventions are used in this manual:

- The names of variables, arrays, input data blocks and files are written using a Courier font.
- The names of all variables and arrays read from the MMV2 input files and used in the grid processing are written in green.
- The names of the **MeshMakerV2.f90** code and of files are written in **boldface**.
- Comments to which special attention needs to be paid are written using *italics*.
- Critically important issues that can severely (and adversely) affect the development of the grid are in red.
- The types (double precision, real, integer, logical character) of input variables and arrays, and the associated formats, are written in blue.
- Reference to real variables in this manual denotes invariably double precision arithmetic, in which all internal computations are conducted.
- Because of the use of free formats for data entry, the *user is strongly advised to enter input character variables within quotation marks*, e.g., 'ft' and not plain ft.
- When the vertical dimension is discretized, by convention the grid layers proceed downward, i.e., from top to bottom.

PAGE LEFT INTENTIONALLY BLANK

2.0. Code Description and Basic Inputs in MESHMAKER V2.0

The space discretization requirements, specification of elements (data block **ELEM**), connections between elements (data block **CONNE**), and the generation of complete grids (file **MESH**) that are needed for TOUGH+ and TOUGH2/3 simulations are described in detail elsewhere [Pruess *et al.*, 1999; 2012; Moridis *et al.*, 2008; Moridis and Pruess, 2014, Jung *et al.*, 2018]. In this section we discuss briefly the conventions used in TOUGH+ v1.5 and TOUGH2/3 codes for entering and processing grid data, as well as the basic **MeshMakerV2.f90** data requirements that provide critical information for the description and definition of the domain under discretization and for the appropriate dimensioning of the code arrays.

2.1. TOUGH+ and TOUGH2/3 Conventions for Processing Discretization Data

Handling of flow geometry data in TOUGH+ [Moridis and Pruess, 2014] is compatible with the TOUGH2 [Pruess *et al.*, 1999; 2012] and TOUGH3 [Jung *et al.*, 2018] input formats and data handling. As in other *integral finite difference* codes [Edwards, 1972; Narasimhan and Witherspoon, 1976], flow geometry is defined by means of a list of volume elements (*grid blocks*), and a list of flow connections between them. This formulation can handle regular and irregular flow geometries in one, two, and three

dimensions. Single- and multiple-porosity systems (porous and fractured media) can be specified, and higher order methods, such as seven- and nine-point differencing, can be implemented by means of appropriate specification of geometric data [Pruess and Bodvarsson, 1983].

In TOUGH+ v1.5 [Moridis and Pruess, 2014], as in TOUGH2 [Pruess et al., 1999; 2012], volume elements are identified by names that consist of a string of either five or eight characters, '12345' or '12345678'; TOUGH3 [Jung et al., 2018] offers more element naming options. These are arbitrary, except that the last two characters (#4 and #5 in 5-character names; #7 and #8 in 8-character names) must be numbers; an example of a valid 5-character element name is 'ELE10', and an example of a valid 8-character name is 'AB00CC23'. Flow connections are specified as ordered pairs of elements, such as such as 'ELE10ELE11' (5-character names) or 'AB00CC23AB00CC24' (8-character names). A variety of options and facilities are available for entering and processing geometric data. As in TOUGH2 [Pruess et al., 1999; 2012], element volumes and domain identification can be provided by means of a data block **ELEME** in the *standard input file*, while a data block **CONNE** can be used to supply connection data, including interface area, nodal distances from the interface, and orientation of the nodal line relative to the vertical. These data are internally written to a disk file **MESH**, which in turn initializes the geometry data arrays used during the flow simulation. It is also possible to omit the **ELEME** and **CONNE** data blocks from the *standard input file*, and provide geometry data directly on a disk file **MESH**.

The MMV2 application involves the **MeshMakerV2.f90** code, and can perform the mesh generation and processing operations discussed earlier. It can generate two-dimensional radially symmetric (r,z) meshes, and one-, two-, and three-dimensional rectilinear (Cartesian) grids in (x,y,z). Multiple-porosity processing for simulation of flow in naturally fractured reservoirs can be invoked by means of a keyword **MINC**, which stands for *Multiple Interacting Continua* (see Pruess et al., 2012; Moridis and Pruess, 2014). The MINC process operates on the data of the primary (porous medium) mesh as provided on disk file **MESH**, and generates a secondary mesh containing fracture and matrix elements with identical data formats on file **MINC**. The file **MESH** used in this process can be either directly supplied by the user, or it can have been generated from an earlier application of the **MeshMakerV2.f90** application.

All the sections in this user's manual provide detailed instructions for preparing the necessary data blocks for the construction of the MMV2 input files and the generation of grids, illustrative examples of which are shown in **Figures 1 to 6**.

2.2. Basic Inputs for the MeshMakerV2.f90 Code

In this section we discuss the basic data needed for the application of the **MeshMakerV2.f90** code. The first data bloc in any MMV2 input file comprises three records (all mandatory), and provide (a) a short description of the problem through an informative title, (b) related to the definition of the domain and the description of the discretization approach and (c) data to allow proper dimensioning of the work arrays and

formatting of the **MESH** file to be generated. These three initial records are discussed in detail below.

Record MESHMAKER.1

The first record of the input file in any **MeshMakerV2.f90** application is the character variable **TITLE**, which includes a header of up to 80 characters and is read using [FORMAT\(A80\)](#). This record is necessary for any MMV2 grid processing simulation to begin.

Record MESHMAKER.2

The following variables are read in MESHMAKER.2 using a [free format](#):

coordinates, MaxNum_Elem, Longest, ElemNameLength

These parameters are defined as follows:

coordinates

Character variable of variable length denoting the coordinate system used in the geometric description of the domain. It may have one of two values (the boldface part of these values is necessary and sufficient):

= '**C**artesian': This option indicates that the region is defined geometrically in terms of Cartesian coordinates.

= '**C**ylindrical': This option indicates that the region is defined geometrically in terms of cylindrical coordinates.

Max_NumElem

Integer denoting the maximum number of elements in the grid.

Longest

Integer indicating the maximum expected number of subdivisions along any of the coordinates in the grid under construction.

ElemNameLength

Integer variable defining the number of characters in the element names. It may have the values of either 5 or 8. **ElemNameLength** = 8 is usually (but not exclusively) used for very large grids involving > 500,000 to millions of cells. If **ElemNameLength** ≠ 8, it is internally reset to the default (= 5).

Record MESHMAKER.3

The following variables are read in MESHMAKER.3 using a [free format](#):

LengthUnits, media_by_number, FracPreparation, FormatType

These parameters are defined as follows:

LengthUnits

Character variable specifying the units of the length-related variables in the provided input data. The following values are acceptable options:

'mm', 'cm', 'm', 'km', 'in', 'ft' or 'mi'

indicating meters, millimeters, meters, kilometers, feet, inches and miles, respectively. Note that all dimensions are converted internally into SI units (meters) for the creation of the **MESH** file, which is the standard unit of length used in the TOUGH+ and/or TOUGH2/3 simulations.

media_by_number

Logical variable specifying how the various media (rocks) are to be named. The possible options are: .T. or .TRUE., in which case the media are named by the character variable **name** (see Section 2.2.2., Record MESHMAKER.4.3.1), and .F. or .FALSE., indicating media identification by the number in the sequence of their listing. The **media_by_number** = .T. option **must** be used to develop the grid in the case of fractured media that are to be described by the Multiple Interactive Continua (MINC) concept, and is needed for the creation of an interim **MESH** file. This file will be further processed to create the MINC file to be used in the simulation. When the **media_by_number** = .TRUE. option is invoked and **FracPreparation** = .TRUE. (see next paragraph),

- the unfractured media are identified by a negative number, and
- the fractured media (which **must** be identified by a 'F' or 'f' as the first letter of the value of the region **name** variable, see Section 2.2.2, Record MESHMAKER.3.3.0) are identified by a positive number.

FracPreparation

Character variable specifying that the grid for a fractured medium is to be developed using the MINC approach. For this to occur, the following three conditions **must** be satisfied:

- (a) **FracPreparation** = .TRUE.,
- (b) **media_by_number** = .TRUE. (see previous paragraph), and
- (c) the medium name **must** have a 'F' or 'f' as the first letter of **H_RegionName**.

If the first two conditions are not met, an error message is printed and the gridding process is aborted. If the first two conditions are met but not the third, then the grid for an unfractured system/medium is developed.

FormatType

Character variable (variable length) indicating the format of the data in the **MESH** file to be created. It may have one of two values:

= 'Old': This option creates a **MESH** file that is consistent with the TOUGH+ V1.5 and TOUGH2/3 standard formats for data inputs.

= 'New': This option creates a **MESH** file in which the element and connection data are listed using the NAMELIST format facility available in FORTRAN 95/2003. This is intended for the next versions of TOUGH+.

3.0. Description of Heterogeneity in MESHMAKER V2.0

This section describes the necessary inputs (data blocks) that are needed if the domain involves heterogeneous regions, boundaries and/or excluded regions. If the system has regular geometry and there is no need for the declaration of any of these three types of heterogeneity, the user can ignore/bypass this entire section and only provide the necessary data for domain discretization in either cylindrical (Section 4.1) or Cartesian coordinates (Section 4.2). Alternatively, it is possible to include any and all of the three heterogeneity-related data blocks and to setting the corresponding number of such incidences to the appropriate number (1 for regions and zero for boundaries and exclusions) if a particular type is absent from (or not needed in) the domain.

3.1. Inputs for the Definition of Heterogeneous Regions

These optional inputs provide information that allows the description of heterogeneity within the domain. The assignment of heterogeneity is based on the definition of *regions* using geometrical information that describe the location and extent of these subdomains. Thus, each of the individually defined regions is assigned the properties of a particular medium that will have to be included in the **ROCKS** or **MEDIA** data block in the input file of the subsequent TOUGH+ v1.5 or TOUGH2/3 simulation.

The records in this data block are discussed in detail below:

Record REGION.1 (Optional)

This record includes only the [keyword \(character variable\)](#) '>>>Regions', which is read using [FORMAT\(A15\)](#). If the domain is homogeneous, then there is no need to provide any of the inputs discussed in this section.

Record REGION.2 (Optional – necessary when the REGION.1 record is included)

This record is read using a [free format](#), and includes the single [integer variable](#) [Num_HetRegions](#) (> 0) that describes the number of heterogeneous regions (subdomains) that are to be described by the ensuing data records. The minimum value that [Num_HetRegions](#) can accept is 1, corresponding to a homogeneous system. An error will occur and the execution will stop if [Num_HetRegions](#) < 1.

Because of dynamic dimensioning, there is no upper limit in the value [Num_HetRegions](#) other than that imposed by the value of [MaxNum_Elem](#) (see Record MESHMAKER.2). However, practical considerations may limit the size of [Num_HetRegions](#). Thus, although it is possible to define element-by-element heterogeneity using this approach, this would be a tedious (but not impossible) process. Generally speaking, the most useful application of this facility is in the description of extensive geologic units with distinctly different properties.

Record REGION.3 (Optional – necessary when the REGION.1 record is included)

This record is read using a [free format](#), and includes only the [5-character variable](#) [dominant_medium](#) that provides the name of the dominant (reference) porous medium in the heterogeneous domain. Note that the selection of a medium as [dominant_medium](#) is arbitrary, as there are no restrictions on the extent of its spatial distribution for it to be designated as such.

If [Num_HetRegions](#) = 1, no more data need to be read for this region (its name suffices) and the next item to be read **must be** Record REGION.5. This case is equivalent to a homogeneous system, and the entire heterogeneity-related data block (records REGION to EXCLUSION) may be omitted.

Record REGION.4.1 (Optional – necessary when the REGION.1 record is included and [Num_Regions](#) > 1)

The following variables are read using a [free format](#):

[name, shape](#)

These parameters are defined as follows:

[name](#)

A [5-character variable](#) that provides the name (5 character long) of the region that is about to be described.

shape

A **character variable** (of variable length) describing the shape of the region. It may have one of the following possible values, the meaning of which is self-explanatory (the boldface part of the word is necessary and sufficient):

= '**R**ectangular'

= '**C**ylindrical'

= '**S**pherical'

Any other value will cause the printing of an error message and the cessation of further reading of the data inputs. Note that when **coordinates** = '**C**ylindrical' (see Section 2.2.1, Record MESHMAKER.2), the **shape** = '**R**ectangular' option is unavailable.

Record REGION.4.2a (Optional – necessary when the MESHMAKER.4 record is included, **Num Regions** > 1 and **coordinates** = '**C**artesian')

If **shape** = '**R**ectangular', the following variables are read using a [free format](#):

Xmin, Xmax, Ymin, Ymax, Zmin, Zmax

These variables are defined as follows:

Xmin, Xmax

Real variables indicating the range of the region along the x-axis of the Cartesian coordinate system.

Ymin, Ymax

Real variables indicating the range of the region along the y-axis of the Cartesian coordinate system.

Zmin, Zmax

Real variables indicating the range of the region along the z-axis of the Cartesian coordinate system.

If **shape** = '**C**ylindrical', the following **variables** are read using a [free format](#):

Orientation, Rmin, Rmax, c01, c02, ODmin, ODmax

These variables are defined as follows:

Orientation

Single-character variable indicating the coordinate direction with which the axis of the cylindrical region is aligned. The possible values of this variable are: 'X', 'Y' or 'Z'.

Rmin, Rmax

Real variables indicating the minimum and maximum radius of the cylindrical region in the cartesian coordinate system.

c01, c02

Real variables defining the coordinates of the center of the cylindrical region in the two non-**Orientation** directions.

ODmin, ODmax

Real variables indicating minimum and maximum coordinates of the cylindrical region in the direction defined by Orientation.

If shape = 'Spherical', the following variables are read using a free format:

Rmin, Rmax, x0, y0, z0

These variables are defined as follows:

Rmin, Rmax

Real variables indicating the minimum and maximum radius of the spherical region in the cartesian coordinate system.

x0, y0, z0

Real variables defining respectively the x-, y-, and z-coordinates of the center of the spherical region in the cartesian domain.

Repeat records REGION.4.1 and REGION.4.2a for each one of the Num_Regions regions.

Record REGION.4.2b (Optional – necessary when the MESHPAKER.4 record is included, Num_Regions > 1 and coordinates = 'Cylindrical')

If shape = 'Cylindrical', the axis of the cylindrical region is always aligned with the axis of the cylindrical coordinate system and the following variables are read using a free format:

Rmin, Rmax, ODmin, ODmax

These parameters are defined as follows:

Rmin, Rmax

Real variables indicating the minimum and maximum radius of the cylindrical region in the cylindrical coordinate system.

ODmin, ODmax

Real variables indicating minimum and maximum coordinate of the cylindrical region in the direction defined by Orientation.

If shape = 'Spherical', the following variables are read using a free format:

Rmin, Rmax, c01

These parameters are defined as follows:

Rmin, Rmax

Real variables indicating the minimum and maximum radius of the spherical region in the cylindrical coordinate system.

c01 Real variable defining the coordinate of the center of the spherical region on the axis of the cylindrical coordinate system.

Repeat records REGION.4.1 and REGION.4.2b for each one of the Num_Regions regions.

Record REGION.5 (Optional – necessary when REGION.1 is included)

This record includes only the [keyword \(character variable\)](#) '<<<', which is read using [FORMAT\(A3\)](#) and indicates the end of the REGION.1 data block that describes the grid regions.

3.2. Inputs for the Definition of Boundaries

These optional inputs provide information that describes the outer boundaries of the domain under discretization. The assignment of grid subdomains as boundaries is based on the geometry-based definition of the outer spatial limits of the domain under discretization. Thus, the boundaries are treated as special types of regions (see Section 2.2.2) with the properties of a particular medium that will have to be included in the **ROCKS** or **MEDIA** data block in the input file of the subsequent TOUGH+ or TOUGH2/3simulation. It is possible for a region and one or more boundaries to be described by the same medium (i.e., a medium that has the same name) in the **ROCKS** or **MEDIA** data block.

The records in this data block are discussed in detail below:

Record BOUNDARY.1 (Optional)

This record includes only the [keyword \(character variable\)](#) '>>>Boundaries', which is read using [FORMAT\(A15\)](#). If the domain is confined by no-flow (Newman-type) boundaries, then there is no need to provide any of the inputs discussed in this section.

Record BOUNDARY.2 (Optional – necessary when the BOUNDARY.1 record is included)

This record is read using a [free format](#), and includes the single integer variable [Num_Boundaries](#) (> 0) that describes the number of boundaries that are to be described. When [Num_Boundaries](#) = 0, no more data need be provided and the next item to be read **must be** Record BOUNDARY.4. There is no upper limit in the value of [Num_Boundaries](#) other than imposed by the value of [MaxNum_Elem](#) (see Record MESHMAKER.2).

Record BOUNDARY.3.1 (Optional – necessary when the BOUNDARY.1 record is included and [Num_Boundaries](#) > 1)

The following variables are read using a [free format](#)

[name](#), [bound_type](#), [shape](#)

and are defined as follows:

[bound_type](#)

A [single-character variable](#) of indicating the type of the boundary. It may have one of two values:

- = 'I': This option indicates an *inactive* boundary, the conditions and properties of which are time-invariant.
- = 'Vxx', with xx indicating integer numbers: This option indicates a time-variable boundary.

The result of this designation is reflected in the **ELEME** block, in which **elem_activity** is set to **bound_type** in all the cells corresponding to the boundary defined here. Any other value of the **bound_type** variable causes the program to print an error message and stop execution.

The other variables in this record are defined in a manner entirely analogous to that in Record REGION.4.1.

Record BOUNDARY.3.2a (Optional – necessary when the BOUNDARY.1 record is included, Num_Boundaries > 1 and coordinates = 'Cartesian')

If **shape** = 'Rectangular', the following variables are read using a [free format](#):

Xmin, Xmax, Ymin, Ymax, Zmin, Zmax

If **shape** = 'Cylindrical', the following [variables](#) are read using a [free format](#):

Orientation, Rmin, Rmax, c01, c02, ODmin, ODmax

If **shape** = 'Spherical', the following variables are read using a [free format](#):

Rmin, Rmax, x0, y0, z0

All these variables in this record are defined in a manner entirely analogous to that in Record REGION.4.2a.

Repeat records BOUNDARY.3.1 and BOUNDARY.3.2a for each one of the **Num_Boundaries** boundaries.

Record BOUNDARY.3.2b (Optional – necessary when the BOUNDARY.1 record is included, Num_Boundaries > 1 and coordinates = 'Cylindrical')

If **shape** = 'Cylindrical', the axis of the cylindrical boundary is always aligned with the axis of the cylindrical coordinate system and the following variables are read using a [free format](#):

Rmin, Rmax, ODmin, ODmax

If **shape** = 'Spherical', the following [real variables](#) are read using a [free format](#):

Rmin, Rmax, c01

All these variables in this record are defined in a manner entirely analogous to that in Record REGION.4.2b.

Repeat records BOUNDARY.3.1 and BOUNDARY.3.2b for each one of the **Num_Boundaries** boundaries.

Record BOUNDARY . 4 (Optional – necessary when BOUNDARY . 1 is included)

This record includes only the keyword (character variable) '<<<', which is read using `FORMAT (A3)` and indicates the end of the BOUNDARY . 1 data block that describes the grid boundaries.

3.3. Inputs for the Definition of Excluded Regions

These optional inputs provide information that describes subdomains that need to be removed from a regularly-shaped domain in order to develop a grid that accurately describes the irregular geometry of a system under study, e.g., a natural reservoir with a complex geometry. The definition of grid subdomains as exclusion zones follows the same approach used in the description of zones, but is treated in the opposite manner, i.e., by removing from the grid any element belonging to the excluded subdomains. Note the excluded subdomains need still to be named for internal identification purposes but the use of such a name is unimportant after the subtraction of the corresponding elements from the grid. Thus, excluded region may have any name and not necessarily any of those included in the **ROCKS** or **MEDIA** data blocks of the input files of the TOUGH family of codes..

The records in this data block are discussed in detail below:

Record EXCLUSION . 1 (Optional)

This record includes only the keyword (character variable) '>>>Exclusions', which is read using `FORMAT (A15)`.

Record EXCLUSION . 2 (Optional – necessary when the EXCLUSION . 1 record is included)

This record is read using a [free format](#), and includes the single integer variable `Num_Exclusions` (> 0) that describes the number of boundaries that are to be described. When `Num_Exclusions` = 0, no more data need be provided and the next item to be read **must be** Record EXCLUSION . 4. There is no upper limit in the number of excluded subdomains defined by `Num_Boundaries`.

Record EXCLUSION . 3 . 1 (Optional – necessary when when the EXCLUSION . 1 record is included and `Num_Exclusions` > 1)

The following [character variables](#) are read using a [free format](#):

`name, shape`

The variables in this record are defined in a manner entirely analogous to that in Record REGION . 4 . 1.

Record EXCLUSION . 3 . 2a (Optional – necessary when the EXCLUSION . 1 record is included, `Num_Exclusions` > 1 and `coordinates` = 'Cartesian')

If **shape** = 'Rectangular', the following variables are read using a [free format](#):

Xmin, Xmax, Ymin, Ymax, Zmin, Zmax

If **shape** = 'Cylindrical', the following variables are read using a [free format](#):

Orientation, Rmin, Rmax, c01, c02, ODmin, ODmax

If **shape** = 'Spherical', the following variables are read using a [free format](#):

Rmin, Rmax, x0, y0, z0

All these variables in this record are defined in a manner entirely analogous to that in Record REGION.4.2a.

Repeat records EXCLUSION.3.1 and EXCLUSION.3.2a for each one of the **Num_exclusions** exclusions.

Record EXCLUSION.3.2b (Optional – necessary when the EXCLUSION.1 record is included, Num Exclusions > 1 and coordinates = 'Cylindrical')

If **shape** = 'Cylindrical', the axis of the cylindrical excluded region is always aligned with the axis of the cylindrical coordinate system and the following variables are read using a [free format](#):

Rmin, Rmax, ODmin, ODmax

If **shape** = 'Spherical', the following variables are read using a [free format](#):

Rmin, Rmax, c01

All these variables in this record are defined in a manner entirely analogous to that in Record MESHMAKER.4.2b.

Repeat records EXCLUSION.3.1 and EXCLUSION.3.2b for each one of the **Num_exclusions** exclusions.

Record EXCLUSION.4 (Optional – necessary when EXCLUSION.1 is included)

This record includes only the [keyword \(character variable\)](#) '<<<', which is read using **FORMAT(A3)** and indicates the end of the MESHMAKER.6 data block that describes the grid boundaries.

NOTE-1: By using multiple region, boundary and exclusion data blocks, it is relatively easy to create heterogeneous domains of very complex geometry that can accurately represent complex geology. Every part of a region that is described by a region data set can be superseded by (and reassigned to) the next region data block if the geometry of the latter covers any part of the geometry of the former. The same applies to boundaries. This does not apply to the case of exclusions: once any part of the domain is excluded, this action cannot be reversed.

NOTE-2: This capability creates files that are directly usable by TOUGH+ codes [Moridis and Pruess, 2014], which can differentiate between active and inactive (with constant or time-variable conditions) elements and do not require positioning of inactive elements at

the end of the list of the active elements. Standard TOUGH2 [Pruess *et al.*, 1999; 2012] does not have this capability. Thus, if **MeshMakerV2.f90** is used to prepare the mesh for a standard TOUGH2 or TOUGH3 simulation, the user may need to modify the resulting **MESH** file (either manually or by using an appropriate script) by (a) collecting all elements having the “I” or “Vxx” activity identifiers (see variable **bound_type**), (b) moving them below the last active element in the **ELEME** data block, and (c) setting the volume of the first inactive element to a zero or negative value (see Moridis and Pruess [2014]). No such manipulation is necessary if the boundary elements are described by an extremely large volume (e.g., $1.0\text{E}50$), in which case such elements are included in the Jacobian computations but practically remain time-invariant by virtue of the magnitude of their volume.

PAGE LEFT INTENTIONALLY BLANK

4.0. Domain Discretization and Grid Generation in MESHMAKER V2.0

The following sections describe the discretization data needed for the grid construction options available in **MeshMakerV2.f90**. These options are activated by appropriate keywords. Thus, the keywords '>>>RZ2D' or '>>>RZ2DL' invoke the generation of a one or two-dimensional radially symmetric (r,z) mesh; the keyword '>>>XYZ' initiates generation of a one-, two-, or three-dimensional Cartesian (x,y,z) mesh. Similarly, the keyword '>>>MINC' invokes the generation of the derivative grid of a fractured domain (cylindrical or cartesian) based on the corresponding, earlier-constructed unfractured grid of the same domain.

The meshes generated under keywords '>>>RZ2D', '>>>RZ2DL' or '>>>XYZ' are internally written to the file **MESH**. When the '>>>MINC' option is invoked, the grid in the **MESH** file is processed to yield the corresponding fractured grid of the domain that is stored in the file **MINC**.

4.1. Inputs for Cylindrical Grid Construction

The section describes the preparation of input data files for the various construction options for radial/cylindrical grids in (r,z) , and provides illustrative examples of such files in **Figures 1** to **4**.

The keywords '>>>RZ2D' or '>>>RZ2DL' invoke generation of a radially symmetric mesh. Values for the radii to which the grid blocks extend can be provided by the user or can be generated internally (see below). Nodal points will be placed at the centers of gravity of neighboring radial interfaces. When '>>>RZ2D' is specified, the mesh is generated by columns; i.e., the **ELEME** block lists first the grid blocks at smallest radius for all layers, then the next largest radius for all layers, and so on.

The keyword '>>RZ2DL' generates the mesh by layers; i.e., the **ELEME** block lists first all grid blocks for the top layer from smallest to largest radius, then all elements for the second layer, and so on. Apart from the different ordering of elements, the two meshes for '>>>RZ2D' and '>>>RZ2DL' are identical. The two alternatives provide a convenience to users to implement boundary conditions by way of inactive elements and to initialize grids. The assignment of inactive elements is discussed in Section 3.2. Gravitationally equilibrated pressure and temperature gradients for single columns are easily obtained with '>>>RZ2D', for subsequent assignment to all columns, including vertical boundaries. The '>>RZ2DL' option facilitates implementation of areal (top and bottom layer) boundary conditions, as well as layer-by-layer initializations after a single-column initialization.

The inputs needed for gridding cylindrical systems are as follows:

Record CYLIND. 1 (Mandatory)

The first record in this data block includes only the [keyword \(character variable\)](#) '>>>RZ2D' or '>>>RZ2DL'. This keyword is read using [FORMAT \(A15\)](#).

Record CYLIND. 2 (Mandatory)

The following variables are read using a [free format](#):

[Z_ref](#), [NRTot](#), [NLTot](#)

These variables are defined as follows:

[Z_ref](#) A [double precision variable](#) that provides the reference elevation, i.e., the elevation at the top of the domain.

[NRMax](#) An [integer variable](#) describing the **total number of radii** defined by the discretization of the radial dimension of the system. Note that this is not the total number of radial increments ΔR resulting from the discretization, which is equal to [NRMax](#) - 1.

[NLTot](#) An [integer variable](#) describing the total number of layers defined by the discretization of the axial dimension of the system.

*There is no upper limit in the values of **NLOG** and **NLTot** other than that imposed by the maximum number of the grid elements **MaxNum_Elem** in Record **MESHMAKER.2** (see Section 2.2.1). This record is followed by any number and combinations of the possible options of the datasets that are discussed below.*

Record **RADII.1** (Mandatory)

This record includes the single **5-character keyword** 'RADII' that is read using a **free format** and indicates that a number of radii corresponding to the outer limits of each subdivision of the radial dimension of the domain are to be read. The RADII data block (i.e., Records **RADII.1** to **RADII.2.1**) **must** be the first discretization definition block after Record **CYLIND.2**.

Record **RADII.2** (Mandatory)

This record includes the single **integer variable** **NRAD** (≥ 1) that describes the number of radii that will be read. At least one radius **must** be provided, indicating the inner boundary of the cylindrical mesh.

Record **RADII.2.1** (Mandatory)

This record includes the **double precision array**

RC(i), i = 1, NRAD

that defines the element boundaries in ascending order and is read using a **free format**. There is no upper limit in the value of **NRAD** other than that imposed by the maximum total number of the radial elements **NRTot** in Record **CYLIND.2**.

Record **EQUDR.1** (Optional)

This record includes the single **5-character keyword** 'UNIDR' that is read using a **free format**. It indicates that the ensuing data describe a set of radial increments of uniformly equal size Δr .

Record **EQUDR.2** (Optional, but necessary when Record **EQUDR.1** is present)

In this record, the following variables are read using a **free format**:

NEQU, DR

These are defined as follows:

NEQU An **integer variable** that provides the number of the equally-sized radial increments. There is no upper limit in the value of **NEQU** other than that imposed by the maximum total number of the radial elements **NRTot** in Record **CYLIND.2**.

DR A **real variable** representing the uniform size of the **NEQU** radial increment.

Record LOGAR . 1 (Optional)

This record includes the single 5-character keyword '**LOGAR**ithmetic' and indicates that this data block will introduce logarithmically distributed radial increments, i.e., increments the size of which that increase from one to the next by the same factor (i.e., $\Delta r_{n+1} = f \Delta r_n$). The boldface part of the word is necessary and sufficient. This keyword is read using a [free format](#).

Record LOGAR . 2 (Optional, but necessary when Record LOGAR . 1 is present)

The variables that in this record are

NLOG, RLOG, DR0

are read using a [free format](#) and are defined as follows:

- NLOG** An integer variable that provides the number of the desired logarithmically distributed radial increments. There is no upper limit in the value of **NLOG** other than that imposed by the maximum total number of the radial elements **NRTot** in Record CYLIND . 2.
- RLOG** A real variable representing the desired radius r_{max} of the last of the **NLOG** radii.
- DR0** The reference radial increment Δr_0 (real variable): the first Δr generated will be equal to $f \cdot \Delta r_0$, with f internally determined such that the last increment will bring total radius to **RLOG** = r_{max} . The factor $f < 1$ for decreasing radial increments is permissible. If Δr_0 is set equal to zero, the last increment Δr generated before the keyword **LOGAR** is invoked will be used as default; in such a case, a value of **DR0** = 0 . 0 must be entered.

Repeat the radial discretization data sets RADII, EQU DR and/or LOGAR as needed. There is no upper limit in the number of the radial discretization data sets as long as $\Sigma(\mathbf{NRAD} + \mathbf{NECU} + \mathbf{NLOG}) \leq \mathbf{NRTot}$ (see Record CYLIND . 2).

NOTE: After the first mandatory RADII data block, an arbitrary number of additional blocks RADII, EQU DR, and LOGAR can be specified in any order. It is necessary that **SUM(NRAD) + SUM(NECU) + SUM(NLOG) = NRTot**. Otherwise, an error message is printed and the grid processing stops.

Record LAYER . 1 (The first LAYER datablock is Mandatory)

This record enters the single keyword (5-character variable) 'LAYER' that introduces information on the axial discretization of the domain (layers), and signals closure of radial discretization data inputs. It is read using a [free format](#).

Record LAYER.2 (Mandatory if Record LAYER.1 is present)

This record includes the variables

NLAY, uniform

that are read using a [free format](#) and are defined as follows:

NLAY An [integer variable](#) specifying the number of equally-spaced axial layers in the cylindrical grid. There is no upper limit in the value of **NLAY** other than that imposed by the maximum allowable total number of layers **NLTot** in Record CYLIND.2.

uniform

A [logical variable](#). A value of **uniform** = .TRUE. indicates uniform discretization of the axial dimension of the domain (uniform equally-sized layer thickness).

Record LAYER.3 (Mandatory if Record LAYER.2 is present)

If **uniform** = .TRUE., this record is read using a [free format](#) and includes the single [real variable](#) **HU** that specifies the uniform thickness of the next **NLAY** layers.

If **uniform** = .FALSE., this record (read using a [free format](#)) includes the **NLAY** elements of the [real array](#)

H(i), i = 1, NLAY

that provides the values of the non-uniformly sized increments of the discretization of the axial dimension of the domain.

A minimum of one set of LAYER data (Records LAYER.1 to LAYER.3) is necessary. Repeat the LAYER data sets as needed. *There is no upper limit in the number of the LAYER data sets as long as $\Sigma(\text{NLAY}) \leq \text{NLTot}$ (see Record CYLIND.2).*

Record CYLIND.3

The end of the cylindrical discretization data blocks is denoted by the [keyword \(character variable\)](#) '<<<' read using **FORMAT (A3)**. If fracture-related processing of the grid is required, the MINC data block is read next (see Section 4.3). Otherwise, the data entry from the input file is completed and Record CYLIND.3 **must** be followed by the keyword '<<<END' (or '<<<End' or '<<<end') read using **FORMAT (A6)**.

NOTE: Any number of LAYER datablocks can be entered, but **it is necessary that $\text{SUM}(\text{NLAY}) = \text{NZTot}$** . Otherwise, an error message is printed and the grid processing stops.

```

==> Meshmaker Test MM_R1a: Cylindrical grid      ! Title
'Cylindrical' 1600 400 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old'     ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>RZ2DL
-1000.0d0 303 5                ! Z_ref, NRTot, NLTot
RADII
  2
  1.0e-5    1.0e-3
EQUADR
  100        2.5e-2
LOGAR
  200        2.5e+3    0.0
EQUADR
  1          1.0e-3
LAYER
  5          .TRUE.
  1.0e00                ! HU
<<<
<<<End

```

```

==> Meshmaker Test MM_R1b: Cylindrical grid      ! Title
'Cylindrical' 1600 400 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old'     ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>RZ2DL
-1000.0d0 303 5                ! Z_ref, NRTot, NLTot
RADII
  2
  1.0e-5    1.0e-3
EQUADR
  100        2.5e-2
LOGAR
  200        2.5e+3    0.0
EQUADR
  1          1.0e-3
LAYER
  5          .FALSE.
  1.0e00 1.0e00 1.0e00 1.0e00 1.0e00 ! H(i)
<<<
<<<END

```

Figure 1. Test MM_R1: examples of simple MMV2 input files describing a regular cylindrical domain without heterogeneities, i.e. no regions with different names, no boundaries and no exclusions. Test MM_R1a has equally-sized subdivisions of the z-dimension; the z-dimension in Test MM_R1b is discretized in unequally-sized increments. The discretization of the radial dimension includes all the possible options available in MMV2. The domains in both cases are discretized by layers. Note that all entries beginning with the '!' characters and beyond are comments for the user's convenience, are not read by the code and can be omitted.

```

==> Meshmaker Test MM_R2          ! Title
'Cylindrical' 14000 250 5          ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old'          ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
2                                  ! ==> Num_HetRegions ( = # of heterogeneous media regions)
'TopLL'                          ! Region #1: dominant_medium
'BotLL' 'cylindrical'             ! Region #2: name, shape
0.0e0 1.5e6 80.0e0 90.0e0         ! Rmin, Rmax, Zmin, Zmax
<<<
>>>Boundaries
0                                  ! ==> Num_Boundaries ( = # of boundaries)
<<<
>>>RZ2D
1.0e2 204 15                      ! Z_ref, NRTot, NLTot
RADII
2
0.004 0.05
EQUDR
1 5.0e-2
LOGAR
199 5.0e+2 0.0e0
RADII
2
5.001e2 5.002e2
LAYER
15 .TRUE.
1.0e+0 ! H0
<<<
<<<END

```

Figure 2. Test MM_R2: MMV2 input file for the discretization a heterogeneous cylindrical domain comprising two different regions (media) medium and including no boundaries. The discretization of the radial dimension includes all the possible options available in MMV2. The domain is discretized by columns.

```

====> Input file Test MM_R3 (large heterogeneous grid)
'cylindrical' 15000 30000 5 ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old' ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
6 ! ==> # of heterogeneous media regions
'HydrL' ! Region #1: Name of dominant medium
'Aquif' 'cylindrical' ! Region #2: name, shape
0.0e0 1.5e3 -6.3e1 -4.825e1 ! Rmin, Rmax, Zmin, Zmax
'OverB' 'cylindrical' ! Region #3: name, shape
0.0e0 1.5e3 -3.0e1 0.0e1 ! Rmin, Rmax, Zmin, Zmax
'UndrB' 'cylindrical' ! Region #4: name, shape
0.0e0 1.5e3 -9.4e1 -6.3e1 ! Rmin, Rmax, Zmin, Zmax
'Casng' 'cylindrical' ! Region #4: name, shape
0.0e0 1.0e-1 -4.6e1 -3.0e1 ! Rmin, Rmax, Zmin, Zmax
'Perfo' 'cylindrical' ! Region #5: name, shape
0.0e0 1.0e-1 -5.225e1 -4.6e1 ! Rmin, Rmax, Zmin, Zmax
<<<
>>>Boundaries
2 ! ==> # of boundaries
'TopBB' 'I' 'cylindrical' ! Boundary #1: name, type, shape
0.0e0 1.5e3 -1.0e-2 0.0e0 ! Rmin, Rmax, Zmin, Zmax
'BotBB' 'I' 'cylindrical' ! Boundary #2: tname, type, shape
0.0e0 1.5e3 -9.4e1 -9.3e1 ! Rmin, Rmax, Zmin, Zmax
<<<
>>>RZ2D
0.0e0 91 68 ! Z_ref, NRTot, NLTot
RADII
3
1.00e-3 1.0e-1 2.00e-1
LOGAR
88 3.9894e+2 0.0
LAYER
10 .FALSE.
1.0e-3 6.50e0 5.00e0 5.0e+0 3.0e+0 3.0e+0 2.0e00 2.0e00 1.0e00 1.0e00
LAYER
45 .TRUE.
5.0e-1
LAYER
13 .FALSE.
1.0e00 1.0e00 2.0e00 2.0e00 3.0e+0 3.0e+0 4.0e+0 4.0e+0 5.00e0 5.0e+0 6.00e0
6.0e+0 1.0e-3
<<<
<<<END

```

Figure 3. Test MM_R3: MMV2 input file for the discretization a heterogeneous cylindrical domain comprising six different regions (media) and including two boundaries. The discretization of the vertical dimension includes three different LAYER data sets. The domain is discretized by columns.

```

====> Input file Test MM_R4 (large heterogeneous grid with a void)
'cylindrical' 15000 30000 5 ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old' ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
6 ! ==> # of heterogeneous media regions
'HydrL' ! Region #1: Name of dominant medium
'Aquif' 'cylindrical' ! Region #2: name, shape
0.0e0 1.5e3 -6.3e1 -4.825e1 ! Rmin, Rmax, Zmin, Zmax
'OverB' 'cylindrical' ! Region #3: name, shape
0.0e0 1.5e3 -3.0e1 0.0e1 ! Rmin, Rmax, Zmin, Zmax
'UndrB' 'cylindrical' ! Region #4: name, shape
0.0e0 1.5e3 -9.4e1 -6.3e1 ! Rmin, Rmax, Zmin, Zmax
'Casng' 'cylindrical' ! Region #4: name, shape
0.0e0 1.0e-1 -4.6e1 -3.0e1 ! Rmin, Rmax, Zmin, Zmax
'Perfo' 'cylindrical' ! Region #5: name, shape
0.0e0 1.0e-1 -5.225e1 -4.6e1 ! Rmin, Rmax, Zmin, Zmax
<<<
>>>Boundaries
2 ! ==> # of boundaries
'TopBB' 'I' 'cylindrical' ! Boundary #1: name, type, shape
0.0e0 1.5e3 -1.0e-2 0.0e0 ! Rmin, Rmax, Zmin, Zmax
'BotBB' 'I' 'cylindrical' ! Boundary #2: tname, type, shape
0.0e0 1.5e3 -9.4e1 -9.3e1 ! Rmin, Rmax, Zmin, Zmax
<<<
>>>Exclusions
1 ! ==> Num_ExcRegions
'Voidd' 'spherical' ! Excluded Region #1: name, shape
0.0 2.0 -3.4751E+1 ! Rmin, Rmax, C0
<<<
>>>RZ2D
0.0e0 91 68 ! Z_ref, NRTot, NLTot
RADII
3
1.00e-3 1.0e-1 2.00e-1
LOGAR
88 3.9894e+2 0.0
LAYER
10 .FALSE.
1.0e-3 6.50e0 5.00e0 5.0e+0 3.0e+0 3.0e+0 2.0e00 2.0e00 1.0e00 1.0e00
LAYER
45 .TRUE.
5.0e-1
LAYER
13 .FALSE.
1.0e00 1.0e00 2.0e00 2.0e00 3.0e+0 3.0e+0 4.0e+0 4.0e+0 5.00e0 5.0e+0 6.00e0
6.0e+0 1.0e-3
<<<
<<<END

```

Figure 4. Test MM_R4: the MMV2 input file for the discretization of a variant of Test MM_R3. It involves a heterogeneous cylindrical domain comprising six different regions (media) medium, including two boundaries and having a spherical exclusion with a radius of 2 m, the center of which is located at $z = -34.751$ m on the main axis.

4.2. Inputs for Cartesian (Rectilinear) Grid Construction

The section describes the preparation of input data files for the various construction options for Cartesian (x,y,z) grids, and provides illustrative examples of such files in **Figures 5 to 9**. The presence of the keyword '>>>XYZ' causes the creation of one-, two-, or three-dimensional Cartesian (x,y,z) meshes, as specified by the appropriate inputs. Note that the space discretization in the TOUGH family of codes is based on the integral finite difference method [Edwards, 1972; Narasimhan and Witherspoon, 1976], which is a member of the finite volume family of methods. This being the case, the use of discretized finite volumes makes every mesh a 3-dimensional one. Thus, a 1-D mesh indicates discretized element volumes aligned on the same axis and a 2-D mesh denotes discretized element volumes arranged on a plane.

The inputs needed for gridding Cartesian systems are as follows:

Record XYZ.1 (Mandatory)

The first record in this data block includes only the [keyword \(character variable\)](#) '>>>XYZ' that invokes generation of a Cartesian (rectilinear) mesh. This keyword is read using a [free format](#).

Record XYZ.2 (Mandatory)

The following variables are read using a [free format](#):

DEG, X_ref, Y_ref, Z_ref

These variables are defined as follows:

DEG [Real variable](#) – the angle (in degrees) between the y-axis and the horizontal. The default (**DEG** = 0.0e0) corresponds to horizontal y- and vertical z-axis. The x-axis is always horizontal.

X_ref, Y_ref, Z_ref
[Real variables](#) – the reference x-, y-, and z-coordinates at the origin of the 3-D axes of the Cartesian system.

Record XYZ.3 (Mandatory)

This record includes the variables

NXtot, NYTot, NZTot

that are read using a [free format](#) and are defined as follows:

Nmtot, m = X, Y, Z
[Integer variables](#) – the total number of subdivisions of the x-, y- and z-dimensions of the domain. The actual number (the sum of the **NO** subdivisions, see next record) *must* be equal with the total in any directions, i.e., $\Sigma(\mathbf{NX}) = \mathbf{NXTot}$, $\Sigma(\mathbf{NY}) < \mathbf{NYTot}$ and

$\Sigma(NZ) < NZTot$. Otherwise, an error message is printed and the grid processing stops.

Record XYZ . 4 (Mandatory)

This record includes the variables

NTYPE, **NO**, **uniform**

that are read using a [free format](#) and are defined as follows:

NTYPE A [single-character variable](#) that can assume one of the values 'NX', 'NY' or 'NZ', specifying grid increments in the *x*-, *y*-, or *z*-direction, respectively.

NO [Integer](#) variable specifying the number of increments on the axis defined by **NTYPE**.

uniform

A [logical variable](#). A value of **uniform** = .TRUE. indicates a **NO** number of equally-sized increments on the axis defined by **NTYPE**.

Record XYZ . 5 (Mandatory)

If **uniform** = .TRUE., this record is read using a [free format](#) and includes the single [real variable](#) **DLU** that specifies the uniform size of the next **NO** increments in the direction defined by **NTYPE**.

If **uniform** = .FALSE., this record (read using a [free format](#)) includes the **NO** elements of the [real array](#)

DEL(i), *i* = 1, **NO**

that provides the values of the non-uniformly-sized increments on the axis defined by **NTYPE**.

Record XYZ . 6 (Mandatory)

The end of the XYZ data block is denoted by the [keyword \(character variable\)](#) '<<<', which is read on the next line after the end of the last Record XYZ . 5 using [FORMAT \(A15\)](#), and **must be followed by the keyword '<<<END' (or '<<<End' or '<<<end') on the next line if the grid processing is completed** and further MINC-processing of the grid is not needed. If such processing is required, the MINC data block is read next (see Section 2.2.5).

*A minimum of one set of data (Records XYZ . 4 and XYZ . 5) for each one of the (x,y,z) axes of a Cartesian system is necessary, with $NO \geq 1$ for each **NTYPE** = 'NX', 'NY' and 'NZ'. There is no upper limit in the number of the XYZ . 4 and XYZ . 5 data sets as long as $\Sigma(NO) \leq \text{MaxNum_Elem}$ (see Record MESHMAKER . 2). The *x*-, *y*-, and *z*-discretization data can be entered in arbitrary order.*

```

Meshmaker Test MM_C1a
'Cartesian' 620 11 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old' ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>XYZ
    0.0e0  5.0e01  2.0e2  3.0e2
    6    5    11
NX      6    .TRUE.
20.0
NY      5    .TRUE.
10.0
NZ      11    .FALSE.
    3.45    4.46    6.31    6.44    7.665    3.454    5.555    2.147
    8.45    5.45    8.12
<<<
<<<<End

```

```

Meshmaker test MM_C1b
'Cartesian' 620 11 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old' ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>XYZ
    0.0e0  5.0e01  2.0e2  3.0e2
    6    5    10
NX      6    .TRUE.
20.0
NY      5    .TRUE.
10.0
NZ      10    .TRUE.
2.0e1
<<<
<<<<End

```

Figure 5. Test MM_C1: Examples of simple MMV2 input files describing a regular Cartesian domain without heterogeneities, i.e., no regions with different names, no boundaries and no exclusions. Test MM_C1a has unequally-sized subdivisions of the z-dimension; the z-dimension in Test MM_1b is discretized in equally-sized increments. All entries beginning with the '!' characters and beyond are comments for the user's convenience, are not read by the code and can be omitted.

```

Meshmaker Test MM_C2
'Cartesian' 5105 51 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old' ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
1      ! ==> Num_MedRegions ( = # of heterogeneous media regions)
'BSand'      ! Region #1: dominant_medium (= name of dominant medium)
<<<
>>>Boundaries
1      ! ==> Num_Boundaries ( = # of boundaries)
'OutBB' 'I' 'rectangular' ! Boundary #1: name, bound_type, shape
0.0e0 1.5e3 5.0e2 5.01e2 -2.0e0 0.0e0 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>XYZ
    0.0e0  0.0e01  0.0e2  0.0e2  ! DEG, X_ref, Y_ref, Z_ref
    10 51 10      ! NXTot, NYTot, NZTot
NX      10      .FALSE.
    0.20    0.50    1.0    2.0    3.0    4.0    5.0    6.0
    8.00    10.10
NY      50      .TRUE.
10.0
NY      1      .TRUE.
1.0e-1
NZ      10      .TRUE.
2.0e0
<<<
<<<<END

```

Figure 6. Test MM_C2: MMV2 input file for a regular Cartesian domain described by a single medium and including a rectangular boundary.

```

Meshmaker Test MM_C3
'Cartesian' 5105 51 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old'  ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
1                          ! ==> Num_MedRegions ( = # of heterogeneous media regions)
'BSand'                   ! Region #1: dominant_medium (= name of dominant medium)
<<<
>>>Boundaries
1                          ! ==> Num_Boundaries
'OutBB' 'I' 'rectangular' ! Boundary #1: name, bound_type, shape
0.0e0 1.5e3 5.0e2 5.01e2 -6.0e0 0.0e0 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>Exclusions
1                          ! ==> Num_ExcRegions
'SEcor' 'rectangular'     ! Excluded Region #1: name, shape
56.0e0 80.0e0 0.0e0 100.0e0 -20.0e0 0.0e0 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>XYZ
      0.0e0 0.0e01 0.0e2 0.0e2
    10 51 10
NX      10 .TRUE.
8.0
NY      50 .TRUE.
10.0
NZ      1 .TRUE.
1.0e-1
NZ      10 .TRUE.
2.0e0
<<<
<<<END

```

Figure 7. Test MM_C3 (a variation of Test MM_C2): the MMV2 input file for irregularly-shaped Cartesian domains that involves a single medium, includes a rectangular boundary and its shape is formed by a rectangular exclusion.

```

Meshmaker Test MM_C4
'Cartesian' 8010 20 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old'  ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
1                          ! ==> Num_MedRegions ( = # of heterogeneous media regions)
'BSand'                   ! Region #1: dominant_medium
<<<
>>>Boundaries
1                          ! ==> Num_Boundaries
'OutBB' 'I' 'cylindrical' ! Boundary #1: name, bound_type, shape
'Y' 0.080 0.09 0.10 -0.10 0.0 0.4 ! orientation,Rmin,Rmax,CLoc(1),CLoc(2),ODmin,ODmax
<<<
>>>Exclusions
1                          ! ==> Num_ExcRegions
'Perim' 'cylindrical'     ! Excluded Region #1: name, shape
'Y' 0.09 0.20 0.10 -0.10 0.0 0.4 ! orientation,Rmin,Rmax,CLoc(1),CLoc(2),ODmin,ODmax
<<<
>>>XYZ
      0.0e0 0.0e01 0.0e2 0.0e2
    20 15 20
NX      20 .TRUE.
0.01
NY      15 .TRUE.
0.02
NZ      20 .TRUE.
0.01
<<<
<<<END

```

Figure 8. Test MM_C4: MMV2 input file for a Cartesian representation of a cylindrical domain with a horizontal main axis. The resulting Cartesian domain can be used for the simulation of a horizontal cylindrical homogeneous core and its holding container (defined the cylindrical boundary 'OutBB') when gravity cannot be ignored and its shape is formed by a cylindrical exclusion.

```

Meshmaker Test MM_C5: Input file for a large 3D cartesian grid      ! Title
'Cartesian' 500000 100 5      ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .FALSE. .FALSE. 'Old'      ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
6      ! Keyword denoting heterogeneous subdomains
      ! ==> # of heterogeneous media regions
'HydrL'      ! Region #1: Name of dominant medium
'Aquif' 'Rectangular'      ! Region #2: name, shape
0.0e0 1.5e3 0.0e0 1.5e3 -6.3e1 -4.825e1      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'OverB' 'Rectangular'      ! Region #3: name, shape
0.0e0 1.5e3 0.0e0 1.5e3 -3.0e1 0.0e1      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'UndrB' 'Rectangular'      ! Region #4: name, shape
0.0e0 1.5e3 0.0e0 1.5e3 -9.4e1 -6.3e1      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'WellA' 'Rectangular'      ! Region #5: name, shape
0.0e0 5.0e-2 0.0e0 5.0e-2 -5.4e1 -3.0e1      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'WellA' 'Rectangular'      ! Region #6: name, shape
4.9995e2 5.0e2 4.9995e2 5.0e2 -5.4e1 -3.0e1      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>Boundaries
2      ! ==> # of boundaries
'TopBB' 'I' 'Rectangular'      ! Boundary #1: name, type, shape
0.0e0 1.5e3 0.0e0 1.5e3 -1.0e-2 0.0e0      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'BotBB' 'I' 'Rectangular'      ! Boundary #2: name, type, shape
0.0e0 1.5e3 0.0e0 1.5e3 -9.4e1 -9.3e1      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>XYZ
0.0 0.0 0.0 0.0 ! DEG, X-ref, Y_ref, Z_ref
76 76 77      ! NXTot, NYTot, NZTot
NX      76 .FALSE.
5.0e-2 2.5000e-1 2.8599e-1 3.2716e-1 3.7426e-1 4.2813e-1 4.8976e-1 5.6027e-1
6.4092e-1 7.3319e-1 8.3873e-1 9.5947e-1 1.0976e+0 1.2556e+0 1.4363e+0 1.6431e+0
1.8797e+0 2.1502e+0 2.4598e+0 2.8139e+0 3.2190e+0 3.6823e+0 4.2124e+0 4.8188e+0
5.5125e+0 6.3061e+0 7.2139e+0 8.2524e+0 9.4404e+0 1.0799e+1 1.2354e+1 1.4132e+1
1.6167e+1 1.8494e+1 2.1157e+1 2.4202e+1 2.7686e+1 3.1672e+1 3.1672e+1 2.7686e+1
2.4202e+1 2.1157e+1 1.8494e+1 1.6167e+1 1.4132e+1 1.2354e+1 1.0799e+1 9.4404e+0
8.2524e+0 7.2139e+0 6.3061e+0 5.5125e+0 4.8188e+0 4.2124e+0 3.6823e+0 3.2190e+0
2.8139e+0 2.4598e+0 2.1502e+0 1.8797e+0 1.6431e+0 1.4363e+0 1.2556e+0 1.0976e+0
9.5947e-1 8.3873e-1 7.3319e-1 6.4092e-1 5.6027e-1 4.8976e-1 4.2813e-1 3.7426e-1
3.2716e-1 2.8599e-1 2.5000e-1 5.0e-2
NY      76 .FALSE.
5.0e-2 2.5000e-1 2.8599e-1 3.2716e-1 3.7426e-1 4.2813e-1 4.8976e-1 5.6027e-1
6.4092e-1 7.3319e-1 8.3873e-1 9.5947e-1 1.0976e+0 1.2556e+0 1.4363e+0 1.6431e+0
1.8797e+0 2.1502e+0 2.4598e+0 2.8139e+0 3.2190e+0 3.6823e+0 4.2124e+0 4.8188e+0
5.5125e+0 6.3061e+0 7.2139e+0 8.2524e+0 9.4404e+0 1.0799e+1 1.2354e+1 1.4132e+1
1.6167e+1 1.8494e+1 2.1157e+1 2.4202e+1 2.7686e+1 3.1672e+1 3.1672e+1 2.7686e+1
2.4202e+1 2.1157e+1 1.8494e+1 1.6167e+1 1.4132e+1 1.2354e+1 1.0799e+1 9.4404e+0
8.2524e+0 7.2139e+0 6.3061e+0 5.5125e+0 4.8188e+0 4.2124e+0 3.6823e+0 3.2190e+0
2.8139e+0 2.4598e+0 2.1502e+0 1.8797e+0 1.6431e+0 1.4363e+0 1.2556e+0 1.0976e+0
9.5947e-1 8.3873e-1 7.3319e-1 6.4092e-1 5.6027e-1 4.8976e-1 4.2813e-1 3.7426e-1
3.2716e-1 2.8599e-1 2.5000e-1 5.0e-2
NZ      14 .FALSE.
1.0e-3 7.00e0 5.00e0 4.0e+0 3.2e+0 2.5e+0 2.0e00 1.6e00
1.25e00 1.0e00 8.0e-1 6.5e-1 5.0e-1 5.0e-1
NZ      45 .TRUE.
4.0e-1
NZ      18 .FALSE.
5.0e-1 5.0e-1 6.5e-1 8.0e-1 1.0e0 1.0e0 1.25e0 1.6e00 2.0e0 2.5e0 3.2e0
3.0e0 4.0e0 4.0e0 5.0e0 6.0e0 8.0e0 1.0e-3
<<<
<<<End

```

Figure 9. Test MM_C5: An example of a MMV2 input file for the creation of a large Cartesian 3D grid with six heterogeneous regions and two different boundaries.

4.2. Inputs for MINC Processing for Fractured Media

The section describes the preparation of input data files for the various construction. This is the third grid construction options available in **MeshMakerV2.f90**. It is applicable to fractured media and is invoked by the '>>>MINC' keyword, which calls a substantially enhanced version of the GMINC program [Pruess, 1983] to sub-partition a primary porous medium mesh into a secondary mesh for fractured media, using the method of *Multiple Interacting Continua* [Pruess and Narasimhan, 1982; 1985].

The MINC-processing operates on the data in a file **MESH** that is either pre-existing or is created directly from the inputs of **MeshMakerV2.f90** when the '>>>RZ2D', '>>>RZ2DL' or '>>>XYZ' data blocks are followed by the '>>>MINC' datablock. MINC-processing creates the external file **MINC**, which is renamed **MESH** for a subsequent simulation involving flow and transport through the fractured domain.

The inputs needed for gridding Cartesian systems are as follows:

Record MINC . 1 (Mandatory)

The first record in this data block includes only the [keyword \(5-character variable\)](#) '>>>MINC' that invokes post-processing of a primary porous medium mesh from a previously developed file **MESH**. This keyword is read using [FORMAT \(A15\)](#). MMV2 allows MINC-processing of elements of any medium (region) and boundary (i.e., of any active or inactive elements) [whose name begins with 'F' or a 'f'](#), and enables seamless and physically correct communication between fractured and unfractured media in the domain.

Record MINC . 2 (Mandatory)

The following variables are read using a [free format](#):

[ProxType](#), [connectivity](#)

These variables are defined as follows:

[ProxType](#)

A [five-character variable](#) for selecting one of the following six different proximity functions provided in MINC [Pruess, 1983].

= '1-D': A set of plane parallel infinite fractures with matrix block thickness between neighboring fractures equal to [PAR \(1 \)](#) (see Record MINC . 3)

= '2-D': Two sets of plane parallel infinite fractures, with arbitrary angle between them. Matrix block thickness is [PAR \(1 \)](#) for the first set, and [PAR \(2 \)](#) for the second set (see Record MINC . 3).

= '3-D': Three sets of plane parallel infinite fractures at right angles, with matrix block dimensions of [PAR \(1 \)](#), [PAR \(2 \)](#), and [PAR \(3 \)](#), respectively -- see Record MINC . 3.

- = 'STANA': Average proximity function for rock loading of the Stanford large reservoir model [Lam et al., 1988].
- = 'STANB': Proximity function for the five bottom layers of the Stanford large reservoir model.
- = 'STANT': Proximity function for the top layer of the Stanford large reservoir model.

connectivity

A **five-character word** for selecting the treatment of global matrix flow. It may have one of the following possible values:

- = ' ' (Blank – default): The global flow occurs only through the fracture continuum, while rock matrix and fractures interact locally by means of interporosity flow (*double-porosity* model).
- = 'MMVER': The global matrix-matrix flow is permitted only in the vertical; otherwise like the double-porosity model; for internal consistency this choice should only be made for flow systems with one or two predominantly vertical fracture sets.
- = 'MMALL': The global matrix-matrix flow in all directions; for internal consistency only two continua, representing matrix and fractures, should be specified (*dual-permeability* model).

NOTE: A user wishing to employ a different proximity function other than the options provided through the **ProxType** variable in MINC needs to replace the function subprogram PROX (x) in **MeshMakerV2.f90** with a routine of the form:

```
FUNCTION PROX(x)
  PROX = (arithmetic expression in x)
  RETURN
END
```

It is necessary that PROX (x) be defined even when x exceeds the maximum possible distance from the fractures, in which case PROX = 1. Additionally, when user supply their own proximity function subprogram, the variable **ProxType** must be set equal to '1-D', '2-D', or '3-D', depending on the dimensionality of the proximity function. This will assure proper definition of the innermost nodal distances [Pruess, 1983].

Record MINC.3 (Mandatory)

This record includes the variables

NumCont, NVOL, FromWhere, (PAR(i), i = 1,..., NVOL)

that are read using a [free format](#) and are defined as follows:

NumCont

Integer – the total number of multiple interacting continua (**NumCont** < 36).

NVOL

Integer – the total number of explicitly provided volume fractions (**NVOL** ≤ **NumCont**). If **NVOL** < **NumCont**, the volume fractions with indices **NVOL**+1, ..., **NumCont** will be internally generated, all being equal and chosen such as to yield proper normalization to 1.

FromWhere

Character variable specifying whether the sequentially specified volume fractions begin with the fractures (**FromWhere** = 'OUT') or in the interior of the matrix blocks (**FromWhere** = 'IN').

PAR(i), i = 1, ..., NumParam

Real array that stores the parameters describing the fracture spacing. The parameter **NumParam** is internally generated, and has the values **NumParam** = 1, 2, 3 for **ProxType** = '1-D', '2-D', or '3-D', respectively. **NumParam** = 0 for all other values of **ProxType**, in which case no **PAR** data are needed.

Record MINC . 4 (Mandatory)

This record includes the **real array**

VOL(i), i = 1, NVOL

that is read using a **free format** and which is defined as the array of length **NVOL** that specifies the volume fractions (each one having a value between 0 and 0) of a continuum with index **i** (for **FromWhere** = 'OUT') or index **NumCont**+1-**i** (for **FromWhere** = 'IN'). For **FromWhere** = 'OUT', **i**=1 is the fracture continuum, **i**=2 is the matrix continuum closest to the fractures, **i**=3 is the matrix continuum adjacent to **i**=2, etc. The sum of all volume fractions must not exceed 1.

Record MINC . 5 (Mandatory)

The end of the MINC data block is denoted by the **keyword (character variable)** '<<<', which (a) is read on the next line after the end of Records MINC . 4 using **FORMAT (A3)** and (b) **must be followed by the keyword '<<<END' (or '<<<End' or '<<<end')** on the next line that is read using **FORMAT (A6)**.

The illustrative examples of the sample input files in **Figures 10** and **11** describe the grid construction alternatives for fractured domains that can be described by the various MINC options that are available in MMV2.

NOTE: *If the application of the **MeshMakerV2.f90** aims to process a pre-existing external **MESH** file, the input file must begin with records **MESHMAKER.1** and **MESHMAKER.2** (see Section 2.2.1) as the first data block, followed by the data inputs described in Records **MINC.1** to **MINC.5**. See the following examples for clarification.*

```

+++>Meshmaker Test MM_FC1           ! Title
'Cartesian' 620 11 5                 ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .TRUE. .TRUE. 'Old'             ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
3                                     ! ==> Num_MedRegions ( = # of heterogeneous media regions)
'Frcsv'                               ! Region #1: dominant_medium (= name of dominant medium)
'HyFrc' 'Rectangular'               ! Region #2: name, shape
0.0e0 1.0 0.0e0 15.0 -12.0 0.0      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'Unfrc' 'Rectangular'               ! Region #3: name, shape
3.0e0 4.0 0.0e0 15.0 -12.0 0.0      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>Boundaries                       !
1                                     ! ==> Num_Boundaries ( = # of boundaries)
'FBund' 'I' 'Rectangular'           ! Boundary #1: name, bound_type, shape
4.0e0 5.0 0.0e0 15.0 -12.0 0.0      ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>XYZ
0.0e+0 0.0e0 0.0e0 0.0e0           ! DEG, Xref, Yref, Zref
5 5 3                               ! NXTot, NYTot, NZTot
NX 05 .TRUE.
1.0
NY 05 .TRUE.
3.0
NZ 3 .FALSE.
4.0 4.0 4.0
<<<
>>>MINC
'1-D' 'MMALL'                       ! ProxType, , Connectivity = Blank, 'MMVER' 'MMALL'
2 1 OUT 1.0e1                       ! NumCont, NVOL, FromWhere, ( PAR(m), m=1,NumParam )
.01                                  ! Vol(1,...,NVOL)
<<<
<<<END

```

Figure 10. Test MM_FC1: the MMV2 single input file for (a) the discretization a heterogeneous cartesian domain comprising three different regions (of which one is fractured) and a fractured boundary and (b) the subsequent creation of the corresponding grid for the fractured domain. The primary (unfractured) grid is stored in the file **MESH**, and the fractured grid is stored in the file **MINC**. For the development of the fractured grid, the following are needed: (a) `media_by_number = .TRUE.`, (b) `FracPreparation = .TRUE.`, (c) the names of the fractured subdomains must begin with a 'F' or 'f' (all three denoted by red) (all three denoted by red) and (d) the **MINC** data block (in blue) must be present immediately after the discretization specification data block **XYZ**.

```

==> Meshmaker test MM_FC2      ! Title
'Cartesian' 19000 30 5        ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .TRUE. .TRUE. 'Old'      ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>Regions
6                               ! ==> Num_MedRegions
'Frcsv'                               ! Region #1: dominant_medium
'HyFrc' 'Rec'                        ! Region #2: name, shape
0.0e0 2.5e-3 0.0e0 7.0e1 -2.3e2 -3.0e1 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'OverB' 'Rec'                        ! Region #3: name, shape
0.0e0 6.21e+2 0.0e0 7.0e1 -3.0e1 0.0e0 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'UndrB' 'Rec'                        ! Region #4: name, shape
0.0e0 6.21e+2 0.0e0 7.0e1 -2.6e2 -2.3e2 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'WellP' 'Rec'                        ! Region #5: name, shape
0.0e0 2.5e-3 6.99e1 7.0e1 -2.3e2 -3.0e1 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'WellI' 'Rec'                        ! Region #6: name, shape
0.0e0 2.5e-3 0.0e0 1.0e-1 -2.3e2 -1.30e2 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>Boundaries                    !
3                               ! ==> Num_Boundaries ( = # of boundaries)
'BckBB' 'I' 'Rec'                 ! Boundary #1: name, bound_type, shape
1.179e2 1.21e+2 0.0e0 2.0e2 -2.6e2 0.0e0 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'BotBB' 'I' 'Rec'                 ! Boundary #2: name, bound_type, shape
0.0e0 6.21e+2 0.0e0 2.0e2 -2.61e2 -2.6e2 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'TopBB' 'I' 'Rec'                 ! Boundary #2: name, bound_type, shape
0.0e0 6.21e2 0.0e0 2.0e2 -1.0e-3 0.0e0 ! Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
<<<
>>>XYZ
00. 0.0e0 0.0e0 0.0e0           ! DEG, Xref, Yref, Zref
11 16 25                         ! NXTot, NYTot, NZTot
NX 8 .FALSE.
0.0025e0 0.0075e0 0.09e0 0.9e0 2.0e0 5.0e0 1.0e1 2.0e1
NX 3 .FALSE.
3.0e1 5.0e1 1.0e-3
NY 16 .FALSE.
0.1e0 0.5e0 1.4e0 3.0e0 5.0e0 7.0e0 8.0e0 1.0e1
1.0e01 8.0e0 7.0e0 5.0e0 3.0e0 1.4e0 0.50e0 0.1e0
NZ 1 .FALSE.
1.0e-3
NZ 7 .TRUE.
6.0e00
NZ 11 .TRUE.
1.2e01
NZ 6 .FALSE.
2.4e01 2.4e01 1.2e01 1.2e01 1.4e01 1.0e-3
<<<
<<<END

+++> MINC test; To follow Meshmaker test MM_FC2
'Cartesian' 19000 30 5        ! coordinates, MaxNum_Elem, Longest, ElemNameLength
'm' .TRUE. .TRUE. 'Old'      ! LengthUnits, media_by_number, FracPreparation, FormatType
>>>MINC
'3-D' ' ' '                   ! FracType, Connectivity = Blank, 'MMVER' 'MMALL'
2 1 OUT 1.0e1 1.0e1 1.0e1    ! NumCont, NVOL, FromWhere, ( PAR(m), m=1,NumParam )
.01                          ! Vol(1,...,NVOL)
<<<
<<<END

```

Figure 11. Test MM_FC2: the MMV2 two separate input files needed for (a) the discretization a heterogeneous cartesian domain comprising six different regions (of which only one is fractured) and an unfractured boundary and (b) the subsequent creation of the the corresponding grid for the fractured domain. The primary (unfractured) grid is obtaining by using the first (top) input file and is stored in the file **MESH**; the fractured grid is from the second (lower, in blue and red colors) input file and is stored in the file **MINC**. For the development of the fractured grid, the following are needed in the top input file: (a) `media_by_number = .TRUE.`, (b) `FracPreparation = .TRUE.` and (c) the names of the fractured subdomains must begin with a 'F' or 'f' (all three denoted by red). For the second run, the first two lines (in blue) of the input file must be the same as in the first input file. The same results can be obtained from a single input file if the **MINC** data block (in red) is placed immediately after the end of the **XYZ** discretization datablock in the first input file (see Figure 10).

PAGE LEFT INTENTIONALLY BLANK

Acknowledgements

This work was supported by the Assistant Secretary for Fossil Energy, Office of Natural Gas and Petroleum Technology, through the National Energy Technology Laboratory, under the U.S. Department of Energy, Contract No DE-AC02-05CH11231.

PAGE LEFT INTENTIONALLY BLANK

References

- Edwards, A.L., TRUMP: A Computer Program for Transient and Steady State Temperature Distributions in Multidimensional Systems, National Technical Information Service, National Bureau of Standards, Springfield, VA, 1972.
- Jung Y., G.S.H. Pau, S. Finsterle and C. Doughty, TOUGH3 User's Guide, Version 1.0, Report LBNL-2001093, Lawrence Berkeley National Laboratory, Berkeley, CA (2018).
- Lam, S.T., A. Hunsbedt, P. Kruger and K. Pruess, Analysis of the Stanford Geothermal Reservoir Model Experiments Using the LBL Reservoir Simulator, Geothermics, 17(4), 595- 605, Report LBL-25957 (1988).
- Moridis, G.J., User's Manual of the MESHMAKER V1.5 Code: A Mesh Generator for Domain Discretization in Simulations of the TOUGH+ and TOUGH2 Families of Codes, Report LBNL-1005134, Lawrence Berkeley National Laboratory, Berkeley, CA (2016).
- Moridis, G.J., M.B. Kowalsky and K. Pruess, TOUGH+HYDRATE v1.0 User's Manual: A Code for the Simulation of System Behavior in Hydrate-Bearing Geologic Media, Report LBNL-0149E, Lawrence Berkeley National Laboratory, Berkeley, CA (2008).
- Moridis, G.J. and K. Pruess, User's Manual of the TOUGH+ v1.5 Core Code: A General Purpose Simulator of Non-Isothermal Flow and Transport Through Porous and Fractured Media, Lawrence Berkeley National Laboratory Report LBNL-6869E, August 2014.

- Narasimhan, T.N. and P.A. Witherspoon, An Integrated Finite Difference Method for Analyzing Fluid Flow in Porous Media, *Water Resour. Res.*, 12(1), 57 – 64 (1976).
- Pan, L., User's Information for WinGridder V3.0. Report LBNL-273E, Lawrence Berkeley National Laboratory, Berkeley, CA (2008).
- Pruess, K., GMINC - A Mesh Generator for Flow Simulations in Fractured Reservoirs, Lawrence Berkeley Laboratory Report LBL-15227, Berkeley, CA (1983).
- Pruess, K. and G.S. Bodvarsson, A Seven-Point Finite Difference Method for Improved Grid Orientation Performance in Pattern Steam Floods, *Proceedings, Seventh Society of Petroleum Engineers Symposium on Reservoir Simulation*, Paper SPE-12252, 175 - 184, San Francisco, CA (1983).
- Pruess, K., C. Oldenburg, and G. Moridis, TOUGH2 User's Guide, Version 2.0, Report LBNL-43134, Lawrence Berkeley National Laboratory, Berkeley, CA (1999).
- Pruess, K., C. Oldenburg, and G. Moridis, TOUGH2 User's Guide, Version 2.1, Report LBNL-43134, Lawrence Berkeley National Laboratory, Berkeley, CA (2012).