Modflow User Tools (MUT) Version 1.31
User's Guide

October 2024

Rob McLaren, Young-jin Park, Sorab Panday

Contents

Chapter 1

Introduction

This document describes a new MODFLOW-USG[1] development environment which has these features:

- We refer to it as Modflow User Tools, or MUT for short.
- MUT is designed to work with a modified version of MODFLOW-USG, where a new surface water flow package, called SWF, has been added. Like the Connected Linear Network (CLN) package, the SWF package represents a new domain type that is fully-coupled to the 3D groundwater flow (GWF) domain. There can also be cell-to-cell flows between the SWF and CLN domains. The SWF domain uses the diffusion-wave approach so simulate 2D surface-water flow. We will refer to this new version of MODFLOW-USG as MODFLOW-USG^{Swf} in this manual.
- We currently develop and run it on a MICROSOFT WINDOWS 10-based computing platform, writing software using the INTEL FORTRAN compiler running inside the MICROSOFT VISUAL STUDIO interactive development environment, which includes software version control tools through GITHUB.
- A text-based approach is used for the MUT interface, in which we first develop an input file of instructions that define our MODFLOW-USG^{Swf} project, then run MUT to read it and write a complete MODFLOW-USG^{Swf} data set. MUT also writes output files for TECPLOT, a third-party visualization software package, which provides a 3D graphical visualization tool to review the model numerical mesh and material properties in the data set. In future, MUT could be extended to support other third-party visualization packages, for example the open source program Paraview.
- MUT can post-process a MODFLOW-USG^{Swf} simulation to provide a TECPLOT visualization of temporal model results, including hydraulic heads, saturations, water depths and flow budget data. *If applied to output files which were produced by an earlier version of Modflow, results may be mixed. It is not our intent here to support all existing Modflow packages, many of which have been superceded.*

This document is subivided into these sections:

Chapter 2 Installation and Setup: How to install MUT, MODFLOW-USG^{Swf} and TECPLOT and define MICROSOFT WINDOWS environment variables.

[1] https://www.gsienv.com/software/modflow-usg/modflow-usg/

Chapter 3 Model Build: How to build a MUT input file, produce a MODFLOW-USG^{Swf} compatible data set and TECPLOT compatible output files with MUT, then review the results of the model build with TECPLOT.

Chapter 4 Model Execution and Post-Processing How to run MODFLOW-USG^{Swf}, convert the output to TECPLOT-compatible output files with MUT, then visualize them with TECPLOT.

Chapter 5 Model Verification Examples used to verify the accuracy of MODFLOW-USG^{Swf} models built using MUT.

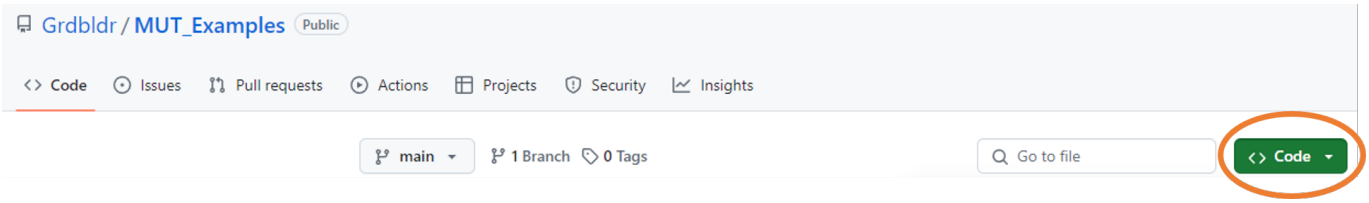
Chapter 6 Illustrative Example An example which illustrates the use of MUT and MODFLOW-USG^{Swf} to simulate variably-saturated, fully-coupled GWFSWF flow in a large-scale watershed.

Chapter 2

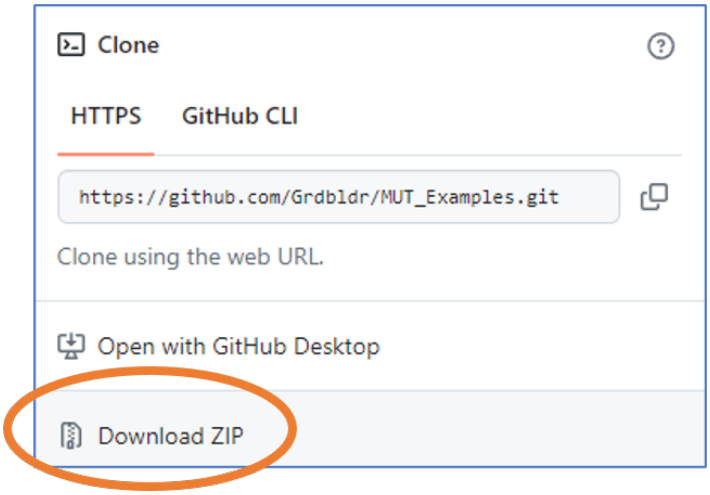
Software Installation and Useage

The first step in the software installation process is to obtain the MUT examples, executables and database files from GITHUB. To do this:

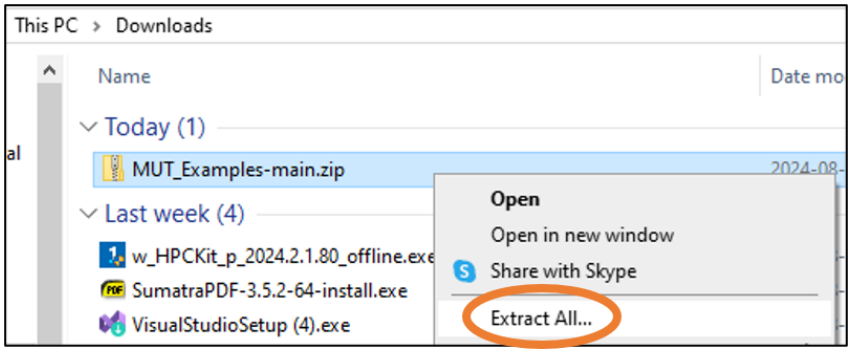
- Click on this link, https://github.com/Grdbldr/MUT_Examples.git, which will take you the MUT_Examples GITHUB page.
- Click on the green 'Code' button.



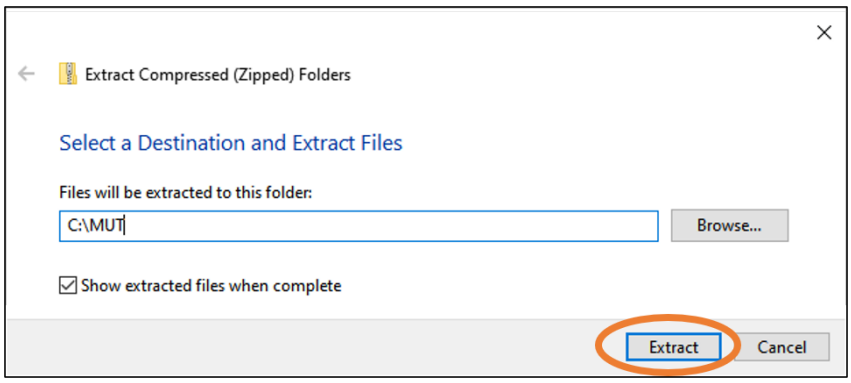
- Choose 'Download ZIP' from the drop-down menu.



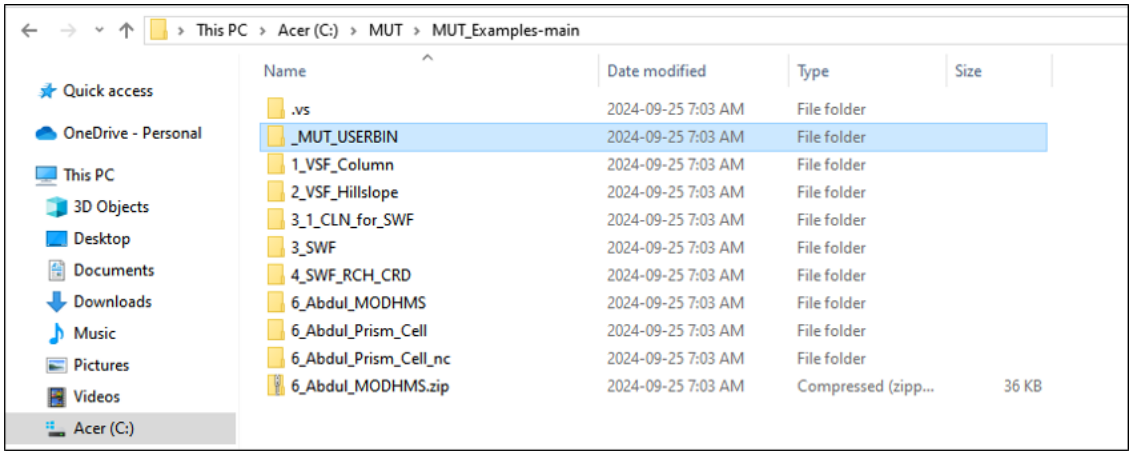
Once the download is complete, the zip file can be found in the MICROSOFT WINDOWS Downloads folder. The contents need to be extracted to a local directory by right-clicking on the download file and choosing 'Extract All...' from the drop-down menu:



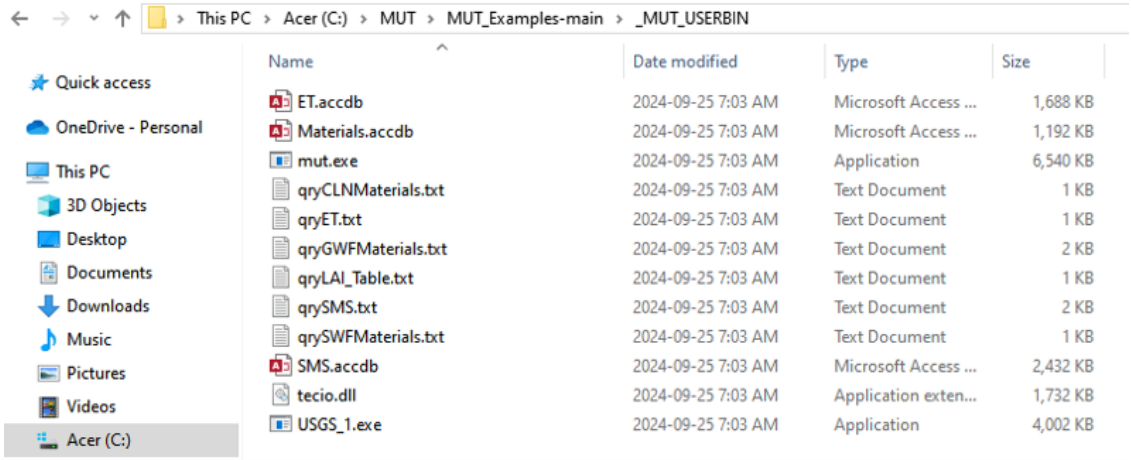
This opens the Extract dialogue, where you are free to choose a different drive and folder to store the extracted files. Here we changed the destination folder to `C:\MUT`. Click the 'Extract' button:



The extracted contents can now be found in the specified destination folder:



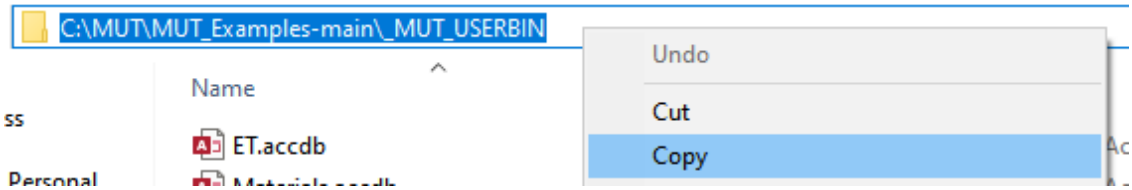
This folder contains a subfolder called `_MUT_USERBIN`, which contains the following files:



These include the executable and supporting files for MUT and the executable for MODFLOW-USG^{Swf}.

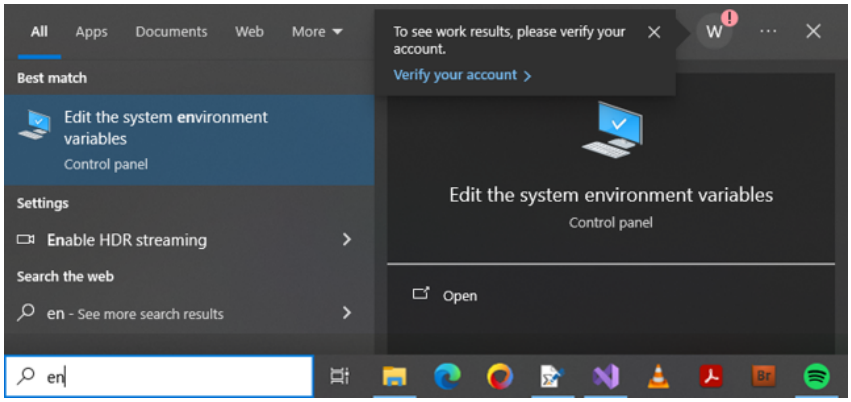
Before you run MUT for the first time, you need to define a windows environment variable called `USERBIN`, which contains the path to the `_MUT_USERBIN` folder, and modify the existing `PATH` variable.

First, highlight the path by clicking on it in the File Explorer window, then copy it by pressing CTRL-C or by right-clicking and choosing 'Copy' from the drop-down menu, as shown here:

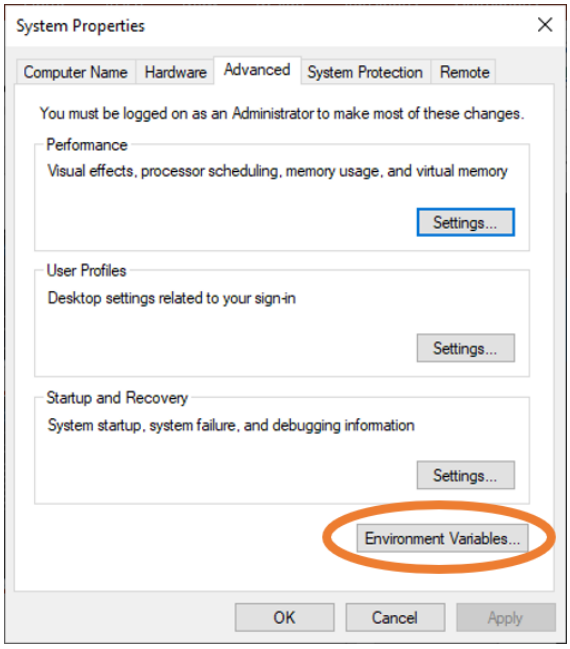


To define the environment variables:

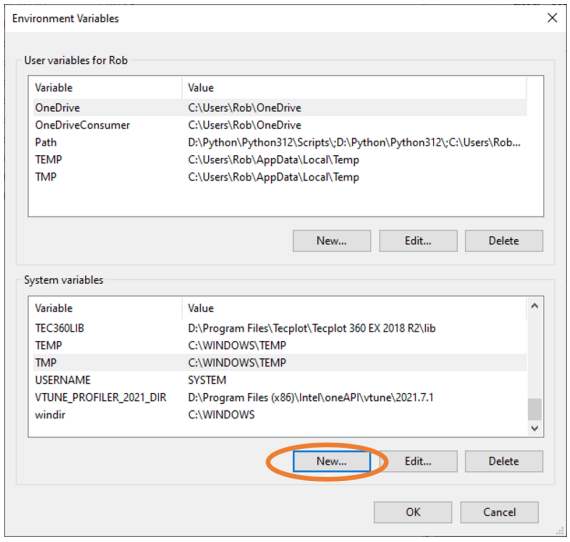
- Type the string 'en' in the windows taskbar search field and open the 'Edit the system environment variables' dialogue:



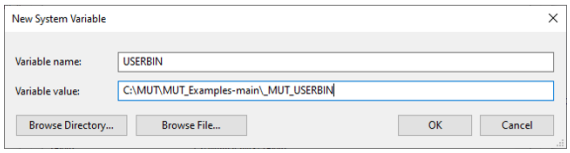
- Click on the 'Environment variables...' button at the bottom of the dialogue:



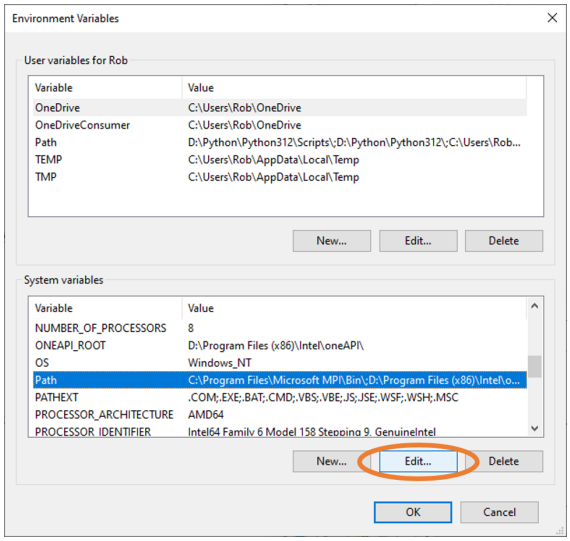
- Click on the 'New' button:



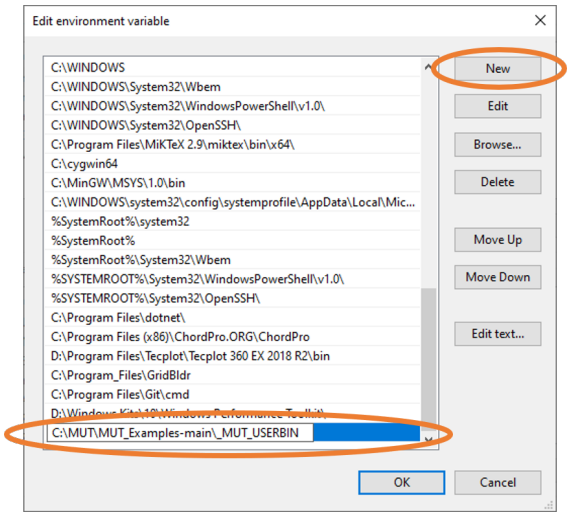
- Add a new variable named `USERBIN` and define the variable value by pasting in the path copied earlier, then click the 'OK' button:



- Add the path to `USERBIN` to the existing `Path` variable. First, choose `Path`, then click the 'Edit...' button:



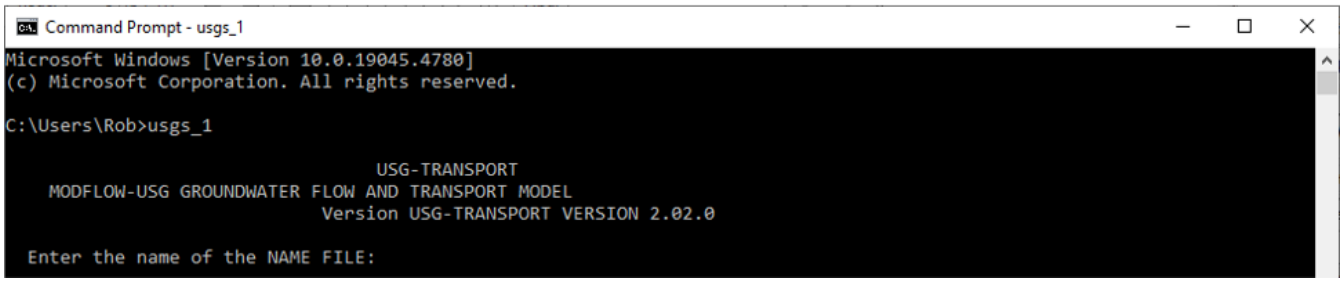
- Click the 'New' button and paste in the path copied earlier, then click the 'OK' button:



You should now be able to run MUT and MODFLOW-USG^{Swf} from the command prompt. To test this, start a new command prompt, then type `mut`, you should see the MUT header:

Type `ctrl-C` to stop the program.

Run MODFLOW-USG^{Swf} by typing `usgs_1`. You should see the MODFLOW-USG^{Swf} header:



Type `ctrl-C` to stop the program.

If this is not the case, check the definitions of the `USERBIN` and PATH variables. If they are correct, you may need to re-boot your computer and try again.

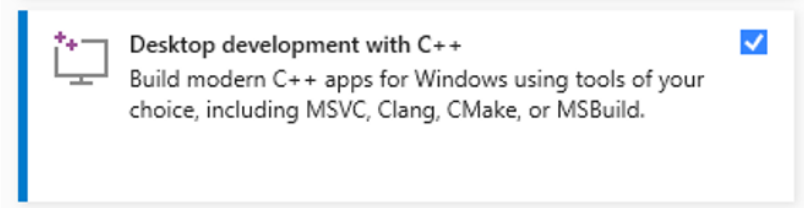
A licensed version of TECPLOT can be obtained from https://tecplot.com/products/tecplot-360/. They have a free 30-day trial option for those who want to assess the software before purchase. They also offer educational discounts.

Those of you who are just interested in running the MUT and MODFLOW-USG^{Swf} programs have completed the required software installation tasks and can proceed to Chapter 3, **Model Build**.

Those who want to view and possibly modify and re-compile the source code for MUT and MODFLOW-USG^{Swf} should proceed with these instructions for setting up your MICROSOFT WINDOWS programming environment.

As was stated earlier, we use and recommend MICROSOFT VISUAL STUDIO and INTEL FORTRAN. You should install MICROSOFT VISUAL STUDIO before INTEL FORTRAN, which will then be automatically integrated into MICROSOFT VISUAL STUDIO.

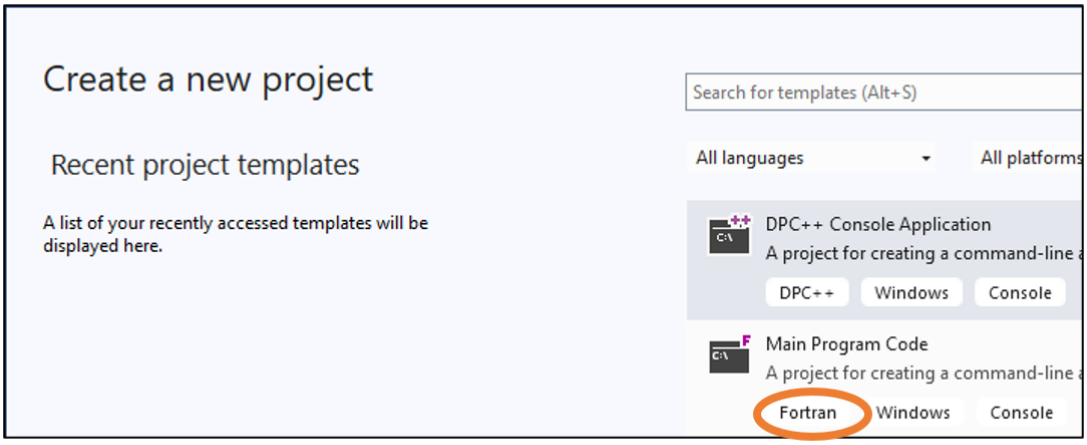
A free version of the latest MICROSOFT VISUAL STUDIO (currently 2022) can be obtained from https://visualstudio.microsoft.com/vs/community/. Once you are on the site just click the `Download` button. This will download a file (e.g. `VisualStudioSetup.exe`) which can be run to install MICROSOFT VISUAL STUDIO. If you already have a version of MICROSOFT VISUAL STUDIO, you can choose to keep your old version and add the latest version. When you come to the installation options 'Workloads' page, be sure to check the option for `Desktop development with C++`, shown here:



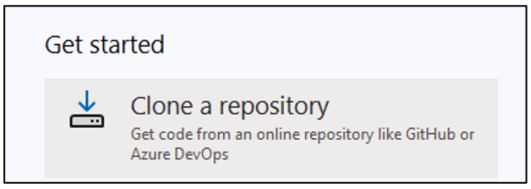
A free version of the latest INTEL FORTRAN compiler can be obtained from https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html. Once you are on the site just click the `Get It Now` button to download the Intel® HPC Toolkit, which includes INTEL FORTRAN. Choose the `Windows` option then the `Offline Installer` option. Now you can either fill in the required information and start the download or choose to `Continue as guest(download starts immediately)`.

This will download a file (e.g. `w_HPCKit_p_2024.2.1.80_offline.exe`) which can be run to install Intel Fortran.

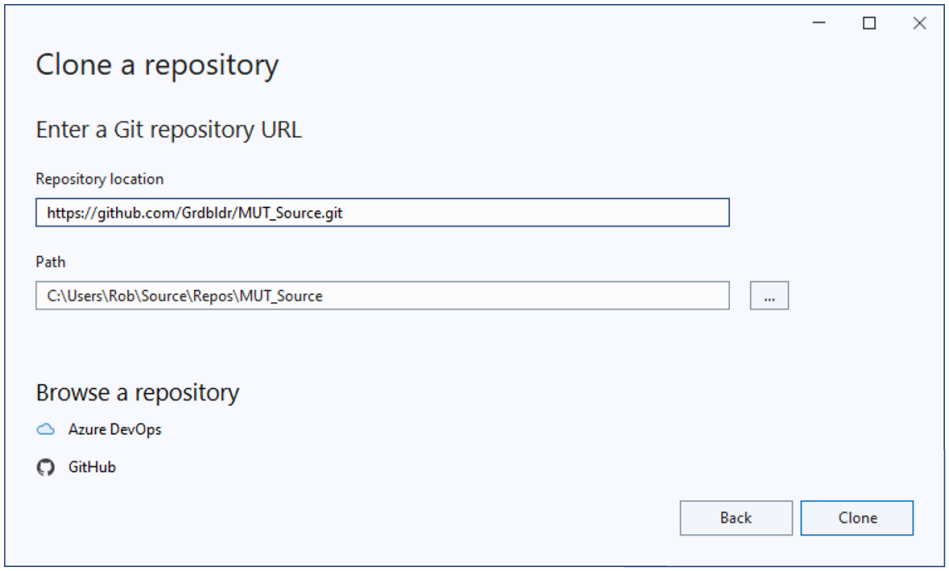
You can check the installation of Microsoft Visual Studio and Intel Fortran by starting Microsoft Visual Studio and choosing `Create a new project`. The window that appears should have links for creating Fortran projects, as shown here:



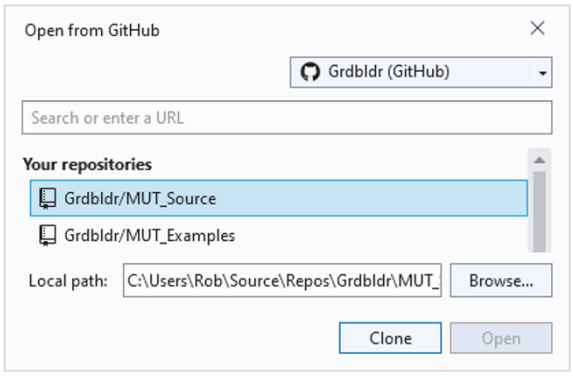
The Mut source files can be obtained from a GitHub repository at https://github.com/Grdbldr/MUT_Source.git. Since GitHub has been integrated into Microsoft Visual Studio we will use it to download the Mut repository. When you start Microsoft Visual Studio choose this option:



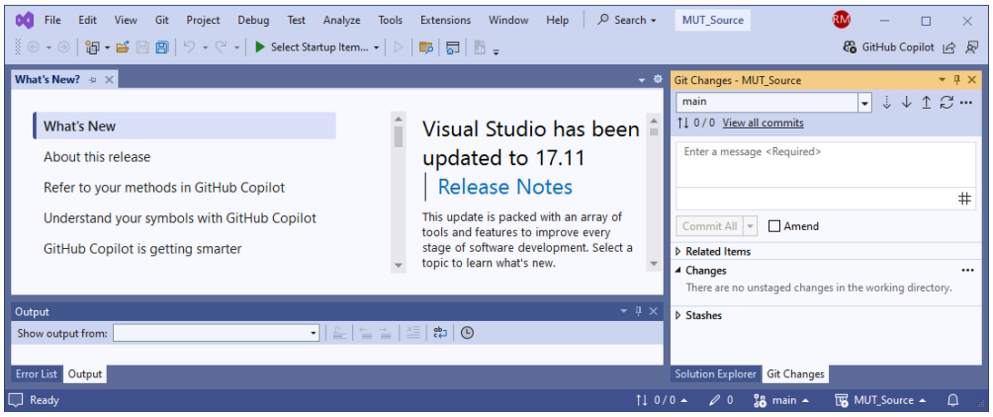
This opens the dialogue box shown below, where you can define the repository location on GitHub and the path to the local repository. You can copy the link from the PDF file by right-clicking on it and choosing `Copy Link Adress`.



Now choose the `GitHub` option under `Browse a Repository` and you will see this dialogue shown below, Choose `Grdbldr/MUT_Source` from the list of repositories then click the `Clone` button.



This shows the Microsoft Visual Studio window after `Grdbldr/MUT_Source` has been cloned. Note the GitHub window on the right side, and information along the bottom about the project:



Details about using GitHub in Microsoft Visual Studio are given in Tutorial A.

The software has been developed and tested under:

- Windows 10
- Tecplot360 EX 2018 R2
- Microsoft Visual Studio Community 2022, Version 17.11.1
- Intel® Fortran Compiler 2024.1

Chapter 3

Mut Execution and Pre-processing

The first step in any model build is to develop a conceptual model, which defines the extent, inflows and outflows, material distributions and physical properties of a hydrogeologic flow system, real or imaginary. The intent of MUT is then to facilitate the production of a set of MODFLOW-USG^{Swf} input files by minimizing the amount of time we spend building and testing it. This chapter describes our current model build workflow, which can provide a sound basis for developing your own personal workflow.

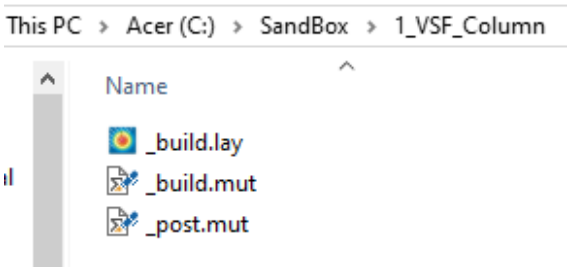
The steps in our model build workflow are:

1. Create a new working folder or copy an existing MUT project folder.
2. Modify the MUT input file (and other input files if necessary) to reflect the new Modflow project.
3. Run MUT to build the new Modflow project, which produces TECPLOT output files for the various Modflow domains (i.e. GWF,SWF and/or CLN) created during the build process.
4. Run TECPLOT and examine the build output files.
5. Repeat steps 2-4 until the new project is defined correctly.

A MUT input file is a plain ascii text file that you can edit with your preferred editor (e.g. Windows Notepad). [1] The MUT input file name must have the extension `mut`, and a prefix of your choice. Examples of valid MUT input file names are `_build.mut` or `good.mut`. Most often, the easiest approach is to copy an existing input file and modify it as required. This helps reduce set-up time and avoid potential errors that are introduced when creating input files from scratch.

To illustrate our model build workflow, we will refer to the various conceptual models developed for our existing suite of verification examples described in Chapter 5. As you read along, we urge you to carry out the steps we describe as we move through the workflow. It is good practice to copy the contents of an existing model to a new location (e.g. copy the folder `MUT_Examples\1_VSF_Column` to `C:\SandBox`) and perform the actions yourself. If you did so, your working directory would look something like this:

[1] Our personal favourite editor is WinEdt (https://www.winedt.com/snap.html), which also provides a nice LaTeX document development environment when coupled with the TeX software package MiKTeX. This manual was produced using these word processing tools.

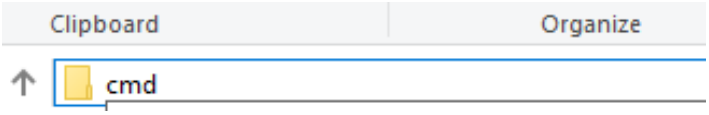


In this example, there are two MUT input files, one for the model build called `_build.mut`, one for post-processing called `_post.mut` (discussed later in chapter 4) and a TECPLOT layout file called `_build.lay` used to visualize the model build results.

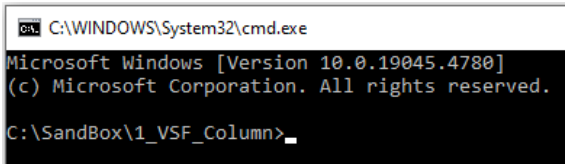
In our preferred workflow, we would start a command prompt in the folder which contains the MUT input file as shown here:

1. Navigate to the folder in File Explorer (e.g. `C:\1_VSF_Column\SandBox`).
2. Click on the path in File Explorer:

3. Replace the existing path with the string 'cmd':



4. Press Enter/Return and you will see a command prompt rooted at the input folder:



When you run MUT it will try to obtain a prefix in the following order:

1. **From a command line argument:** At the command prompt, MUT checks for the presence of a command line argument. For example, typing this:

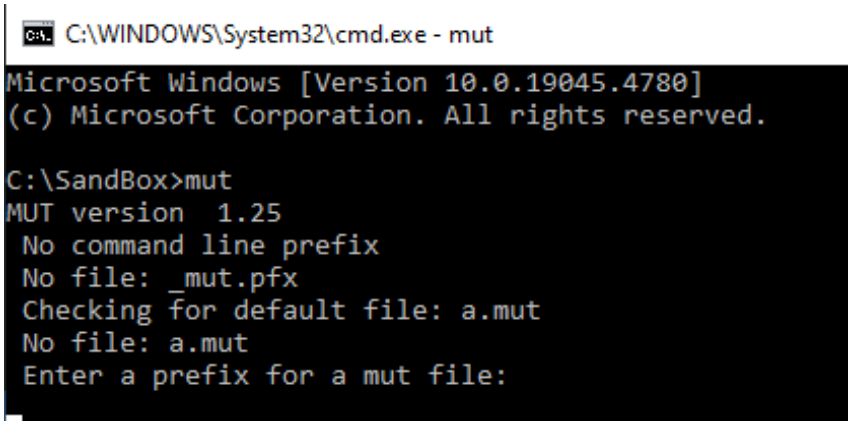
   ```
   mut MyInput
   ```

   would cause MUT to process the input file `MyInput.mut`.

2. **From a prefix file:** If there is no command line argument, MUT checks for the presence of the file `_mut.pfx` in the folder. If present, MUT will read the prefix from it. For example, if the mut file was called `_build.mut` then the file `mut.pfx` would have the single line `_build`.

3. **From the default input file:** If there is no command line argument or prefix file in the folder, MUT checks for the presence of the file `a.mut`. If present in the folder, MUT will use it.

4. **From the keyboard:** If none of these methods are successful, MUT will prompt for a prefix as shown here:



So for example, we could run MUT using the input file `_build.mut` by typing:

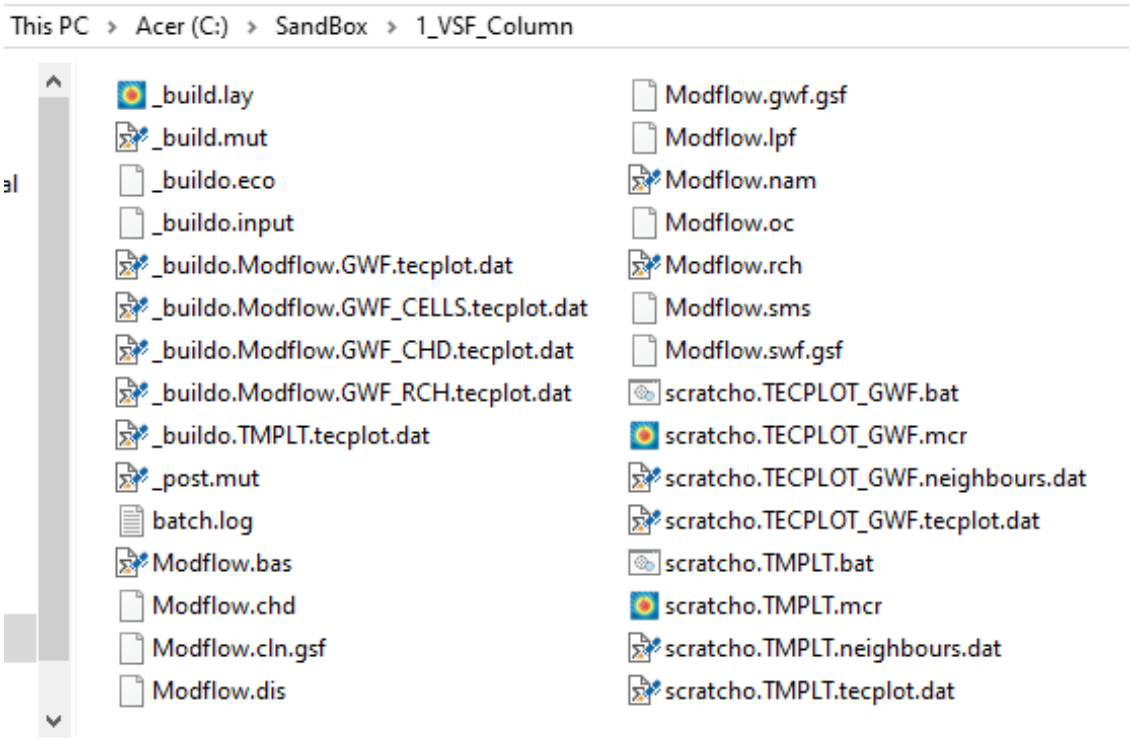
```
mut _build
```

which uses the first method to supply the prefix.

As MUT processes the input file output is written to both the screen and to the file `_buildo.eco` as execution progresses. The first thing written is the MUT version number, then the formal header, which also contains the build date.

Comment lines that are stripped from the input file are echoed to the screen and `_buildo.eco` file and can provide a synopsis of the input file contents.

After MUT finishes, the working folder should look something like this:



Several new output files have been created, of which it may be noted:

- Build output files, which have the prefix `_buildo`, appear near the start of the list if sorted by name.
- TECPLOT output files are indicated by the suffix `.tecplot.dat`.
- Modflow model input files are written using the default prefix `Modflow`, (e.g. `Modflow.nam`, `Modflow.bas` etc.) The prefix can be customized if desired but there are advantages to keeping this 'generic' one, such as portability of post-processing scripts or TECPLOT layout files that follow this generic naming convention.
- Several scratch files (with prefix `scratcho`) are written. These are used for debugging during code development and can be ignored in most cases.
- MUT deletes previously generated output files and writes a fresh set each time it is run. This can prevent confusion that might arise if out-of-date output files were present. [2]
- If the run is successful the last line written will be `Normal exit`, otherwise an error message will be given.

If you open the file `_build.mut` in your preferred text editor and you will see the first couple of lines are comments describing the problem:

```
! Examples\1_VSF_Column:
!   A modflow project of a 1D column generated from a simple 2d rectangular mesh
```

Comments begin with an exclamation point character: ! . MUT creates a clean copy of the input file called *prefix*`o.input` by removing all comment lines, then processes that file to build the model. The cleaned input file contains MUT instructions, which may require data in the form of numbers (e.g. parameter values) or alphanumeric strings (e.g. file names).

The first instruction in the input file begins the model build:

build modflow usg

This is a *subtask* that defines the components of the MODFLOW-USG^{Swf} model such as:

- Units of length and time
- Numerical model meshes
- Material properties
- Boundary conditions
- Solver parameters
- Timestepping,stress periods and output control

[2] For example, if we define a recharge boundary condition, MUT will create the file *prefix*`o.Modflow.SWF_RCH.Tecplot.dat` which shows the locations and recharge values assigned to Modflow cells. If we then removed the recharge condition from the input file, but did not delete this output file, we may assume the recharge condition still applies.

Subtasks have their own unique set of instructions, which are read and processed until an end instruction is encountered. We suggest appending the subtask name to the end instruction, which makes debugging easier when subtasks are nested:

end build modflow usg

We will use the formatting convention shown above when documenting new instructions:

- Heavy upper and lower lines frame the instruction documentation.
- The instruction name is presented in a large sans-serif font.
- Data inputs, if required, are presented and described in a numbered list.
- General notes about instruction usage are presented.
- In the case of a subtask instruction, a suggested end instruction is presented. The first three non-blank characters must be the string end, but the rest is optional.

3.1 Defining the Units of Length and Time

By default, MUT uses meters and seconds as the units of length and time respectively.

The instruction units of length is used to change the default value:

units of length

1. **$_units** The desired units of length: feet, meters or centimeters.

So, for example, to use units of centimeters instead of meters, we would put the string centimeters in the input file, which would then be assigned to the string variable **$_units**.

This instruction requires one line of input, which in this case is a variable named **$_units**. When naming input variables in the command description, the following conventions will be used:

- Alphanumeric string variable names will begin with the string '**$_**'
- Real number (i.e. containing a decimal point) variable names will begin with the string '**R_**'
- Integer number (i.e. *not*a containing decimal point) variable names will begin with the string '**I_**'

The instruction units of time is used to change the default value:

units of time

1. **$_units** The desired unit of time: seconds, minutes, hours, days or years.

So, for example, to use units of days instead of seconds, we would put the string days in the input file, which would then be assigned to the string variable **$_units**.

MUT converts the string variable **$_units** to its numeric equivalent and passes that to MODFLOW-USG^{Swf} through the variables ITMUNI and LENUNI.

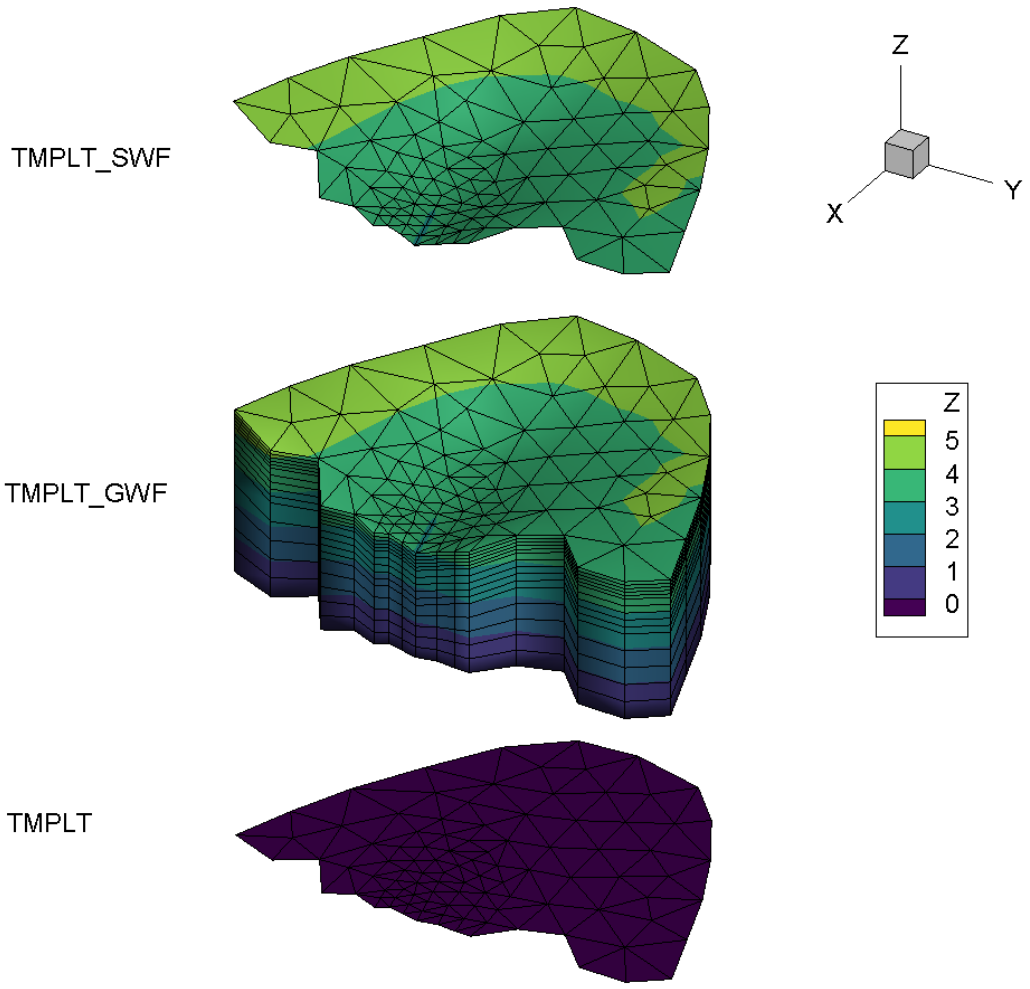
NOTE: *When supplying input data that has units, you must be careful to supply the values in the unit system defined for the model. For example, hydraulic conductivity has units or length/time (L/T), so for the default case the values would be given in units of m/s.*

The supplied databases (see section **??***) have fields that define the length and time units that apply to each database record, in which case* MUT *will convert them to the model defined unit system automatically.*

3.2 Defining the Template Mesh

The next step in the model build workflow is to define a template mesh, which is a 2D finite-element mesh that is used to generate a 3D GWF (and possibly a 2D SWF) finite-element mesh. Below is an example [3] showing an exploded view of a template mesh (bottom image) that was used as a basis for generating finite-element meshs for the GWF (middle image) andSWF (upper image) domains:

[3] This example was generated using the TECPLOT layout file `MUT_Examples\6_Abdul_Prism_Cell\FIG Template Abdul.lay`.



Some key features of this example are:

- The template mesh is assigned an elevation of zero, and only the xy coordinate data are used to define the other domains.
- The GWF domain has been assigned a base elevation of zero, and a variable top elevation.
- The SWF domain has been assigned the same elevation as the GWF domain i.e. they are coincident.

In this example, the template mesh was defined using these instructions:

```
2d mesh from gb
.\gb\grid
```

The instruction 2d mesh from gb, which requires a single line of input, `.\gb\grid`, is documented as shown here:

2d mesh from gb

1. **$_Prefix** The GRID BUILDER [4] dataset prefix, including the path to it.

Given **$_Prefix**, this instruction reads the 2D finite-element grid data and uses it to define the 2D template mesh. **$_Prefix** should contain a relative path to the dataset. Examples of relative paths are:

.\gb\grid The MUT input folder contains a local folder `gb` with the data set prefix `grid`.

..\gb\grid The parent folder to the MUT input folder contains a folder `gb` with the data set prefix `grid`.

C:\gb\grid Absolute path to a drive `C:` with a folder `gb` with the data set prefix `grid`. Absolute paths are not recommended as they may lead to portability issues.

To generate uniform 2D rectangular element template meshes [5] use this instruction:

generate uniform rectangles

1. **R_xl**, **I_nbx** Domain length and number of blocks in the x-direction
2. **R_yl**, **I_nby** Domain length and number of blocks in the y-direction

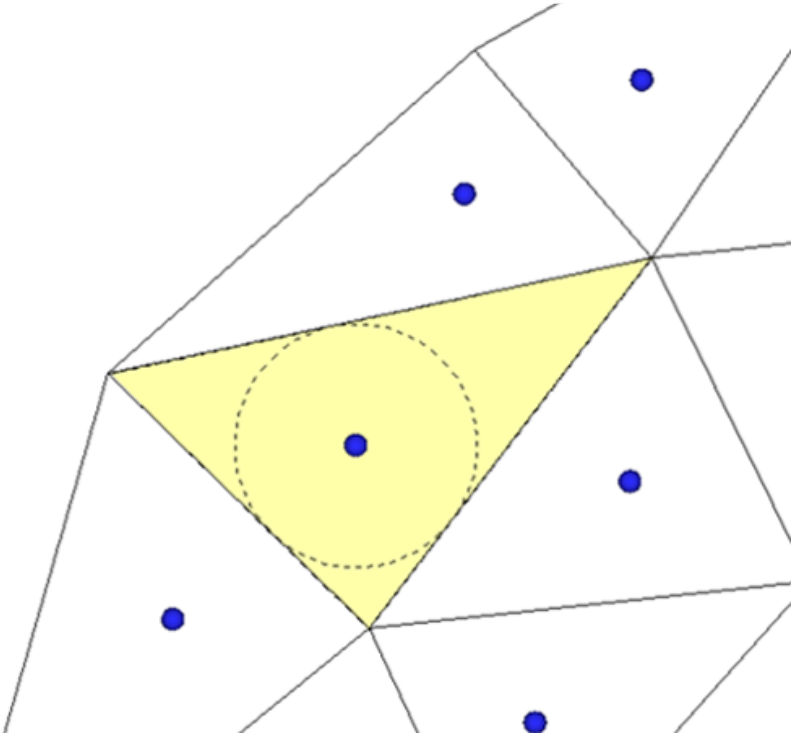
A 2D finite-element mesh composed of uniform rectangular elements will be generated. In this case, the grid is formed by subdividing the domain in the x-direction into **I_nbx** blocks, each of length **R_xl/I_nbx**. The domain is subdivided in a similar fashion in the y-direction, using the other input parameters.

Do we have a working quadtree example to add here as an option.

There are two ways that MODFLOW cell control volumes can be defined from the template mesh. By default, MUT uses a mesh-centred approach as shown here for a triangular-element template mesh:

[4] GRID BUILDER is a legacy 2D triangular finite-element grid generator.

[5] See for example the verification cases `MUT_Examples\1_VSF_Column` or `MUT_Examples\6_Abdul_MODHMS`



Some key features to note are: .

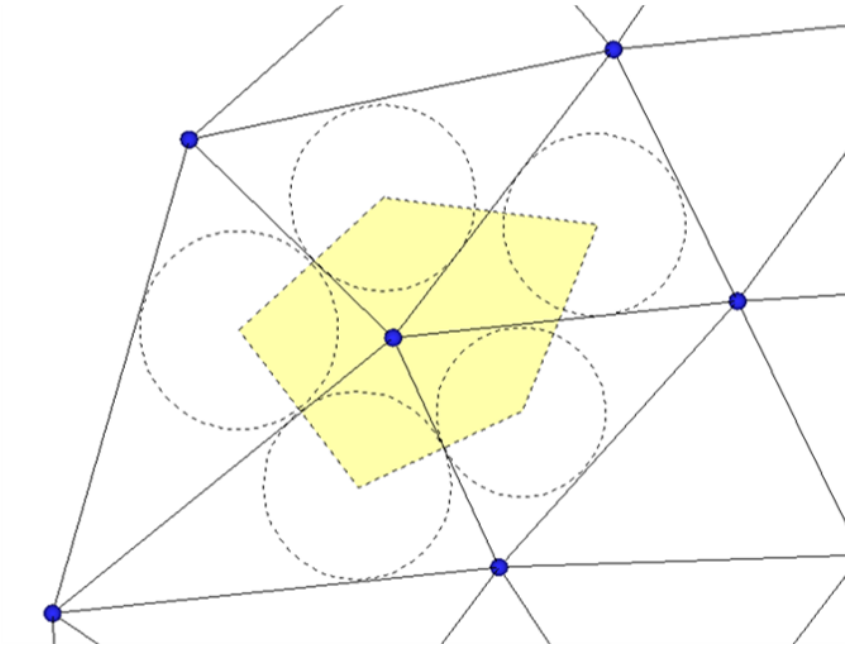
- Inner circles, which are tangent to all three element sides, are defined for each triangular element. An example is shown by the dashed circular line in the yellow-shaded element.
- The blue-filled circles show the locations of the defined MODFLOW cell control volumes.
- The vertical connection area of the cell is defined by the triangular element area (yellow-shaded triangle).
- The horizontal connection length of the cell is defined by the triangular element side length between neighbouring elements.

The mesh-centred approach is similar when using a rectangular-element template mesh, with the rectangular element area and side lengths defining the vertical connection area and horizontal connection length respectively.

To use a node-centred control volume approach, add this instruction *before* defining any GWF or SWF model domains:

```
nodal control volumes
  The node-centered approach will be used to define MODFLOW cell centres instead of the
  default mesh-centered approach.
```

The result of using a node-centred approach is shown here for a triangular-element template mesh:



Some key features to note are: .

- The blue-filled circles show that the locations of the defined MODFLOW cell control volumes are now located at template mesh node locations.
- The vertical connection area of the cell (yellow-shaded polygon) is defined by the contributing area formed by joining the inner circle centres of each element containing the template mesh node.
- The horizontal connection length of the cell is defined by the distance to a neghbouring node.

Should explain how inner circle radius is used for connection length and perpendicular area for triangles.

3.3 Groundwater Flow(GWF) Domain

Currently, every MODFLOW-USG^{Swf} model must contain a GWF domain, which may be reduced to a single layer of very low hydraulic conductivity in cases where GWF flow and interaction with other model domains is to be neglected.

3.3.1 Generating a Layered GWF Domain

A MODFLOW-USG^{Swf} 3D groundwater flow (GWF) domain can be generated from the template using this instruction:

---

generate layered gwf domain
This subtask has instructions that are used to define:

- Element zone numbering scheme
- Top elevation (i.e. z-coordinate)
- Mesh layers and vertical discretization

Subtask instructions will be read and processed until an end instruction is encountered. We suggest appending the subtask name to the end instruction:

end generate layered gwf domain

The construction of the 3D GWF finite-element mesh proceeds from top to bottom. First, we define the top elevation, then add layers one at a time until we reach the base of the domain. By default, element zone numbers will be assigned by layer number. If the template mesh is divided into horizontal patches with unique zone numbers, these can be assigned instead to the 3D GWF mesh [6] using this instruction:

Zone by template
Causes MUT to assign the template mesh element zone number to the corresponding 3D GWF element.

This instruction should appear in the input file at the beginning of the generate layered gwf domain subtask before new layers are added.

3.3.1.1 Defining the Top Elevation

To assign an elevation to the top layer of template nodes use this instruction:

---

top elevation
This subtask defines the elevation (i.e. z-coordinate) of the top layer of nodes in the GWF finite-element template mesh in one of these ways:

[6] The verification example `MUT_Examples\6_Abdul_Prism_Cell` uses this option to define SWF domain zones.

- By assigning a given elevation to all nodes
- By reading variable elevation data from a file
- By interpolating elevation data from a function $z(x)$ where the elevation z varies by the nodes x coordinate.

Once the elevation is defined, an end instruction is required to stop the subtask e.g.

```
end top elevation
```

The top elevation can be defined by one of these instructions:

```
elevation constant
```

1. **R_elev** The elevation **R_elev** will be assigned to all top layer nodes.

```
elevation from gb file
```

1. **$_file** The elevation data in the GRID BUILDER nodal property file named **$_file** will be assigned to the top layer nodes.

The GRID BUILDER nodal property file uses a legacy binary file format. You can develop your own ascii input files and read them using this instruction:

```
elevation from list file
```

1. **$_file** The elevation data in the ascii file named **$_file** will be assigned to the top layer nodes.

Part of a sample list file [7] is shown here:

```
Kriged cell top elevation for layer 1
 4.414571762E+000
 4.415914536E+000
 ...
 4.415914536E+000
```

Some key features of this example are:

- The first line of the file is discarded, and in this case contains a string describing the data.

[7] The verification example `MUT_Examples\6_Abdul_MODHMS` uses an ascii file input to define nodal elevations.

- You must supply a value for each node in the template finite-element mesh.
- The data is read in free format so there can be more than one value entered per line.
- Only the start and end of the file are shown here, with the string '...' replacing the middle portion.

To define the top elevation as a function of x (usually used for cross-sectional models) use this instruction:

elevation from xz pairs

1. **R_x(1)**, **R_y(1)** First x, z coordinate pair.
2. **...**
3. **R_x(n)**, **R_y(n)** nth x, z coordinate pair.

An elevation is calculated for each chosen cell, based on it's x-coordinate location, by interpolating an elevation from the given list of xz-coordinate pairs.

This subtask reads a list of xz-coordinate pairs until an end instruction is encountered e.g.

end elevation from xz pairs

Here is an example showing the use of this instruction [8]:

```
elevation from xz pairs
       0.0, 0.0
    1000.0, 100.0
end elevation from xz pairs
```

Some key features of this example are:

- The two given xz pairs define a line that slopes from $z = 0$ at $x = 0$ to $z = 100.0$ at $x = 1000$. You may supply as many pairs as needed to define the top of your cross-section.
- x coordinates must increase continuously from the top of the list to the bottom.
- the x-range of the supplied pairs should cover the entire x-range of the template mesh.
- For each node in the template mesh, the x coordinate is used to interpolate an elevation (i.e. z value) using the appropriate xz pair.

[8] The verification example `MUT_Examples\1_VSF_Hillslope` uses the elevation from xz pairs pairs instruction to define the top elevation of the cross-sectional domain.

3.3.1.2 Adding Layers

NOTE: The term layers used here should not be confused with the MODFLOW *term of the same name. A* MODFLOW *layer is one cell thick, while a* MUT *layer can be one or more elements thick.*

A MODFLOW-USG^{Swf} model must contain at least 1 layer, and each layer is defined using this instruction:

---

new layer

This subtask adds a new layer to the GWF domain by defining the layer:

- Base elevation
- Vertical discretization

It reads instructions until an end instruction is found e.g.

end new layer

The base elevation is defined using the elevation instructions described on page 25 that are given for the top elevation instruction.

By default, a new layer will be assigned the name '`Layer n`' where n is the current layer number. If you want to assign your own layer name use this instruction:

Layer name

1. **$_layer_name** Layer name.


These names are not currently used in Tecplot output but could/should? be used to create the customlables for zone naming.

By default, MUT will stop and issue a warning message if the computed layer base elevation is greater than or equal to the current layer top elevation. This instruction forces the base to be below the top by a set amount:

Minimum layer thickness

1. **R_MinThick** Minumum thickness value[L].

This instruction causes MUT to enforce a minimum thickness constraint for the current layer. At nodes where the computed layer base elevation is greater than or equal to the current top elevation, **R_MinThick** will be subtracted from the current top elevation to get the base elevation.

By default, a new layer will not be subdivided vertically unless one the following two instructions is issued. The first creates a uniform subdivision:

Uniform sublayering

1. **I_nsublayer** Number of sublayers.

This instruction divides the layer vertically into **I_nsublayer** elements, which will each have the same element height, equal to the top elevation minus the current base elevation divided by **I_nsublayer**.

This instruction creates a non-uniform subdivision:

Proportional sublayering

1. **I_nsublayer** Number of proportional sublayers.
2. **R_sub_thick(i),i=1,I_nsublayer** Proportional thicknesses in order from top to bottom.

This instruction can be used if you want to refine the GWF domain mesh vertically, for example, in the active zone with the SWF domain the ground surface in the .

It is important to understand that the variable **R_sub_thick** is not a true thickness, but is instead a relative thickness, which is used along with the layer thickness to determine the element heights in the current column.

For example, these instructions:

```
Proportional sublayering
   3
   0.1
   1.0
   10.0
end
```

would subdivide the current layer vertically into three elements, between the current base and top elevation, with element height proportions of .1, 1 and 10 from top to bottom.

This instruction is most often used to define a layer of uniform thickness relative to an uneven top elevation:

Offset base

1. **R_value** Thickness value (L) by which to offset the layer base elevation.

This instruction causes the elevation of the base of the layer to be offset vertically by the given value. This can be used to create a surface a given distance below another surface.

For example, these instructions:

```
        top elevation
            elevation from list file
            elev.list
        end top elevation

        new layer
            uniform sublayering
            3

            elevation from list file
            elev.list

            offset base
            -1.0

        end new layer

    end generate layered gwf domain
```

create a layer with a top elevation 1 metre below the elevation defined in the raster file `elev.list`:

Need to check sign on offset base *input*

3.3.1.3 Cell Connection Properties

GWF cell to GWF cell connection properties are automatically defined by MUT *based on control-volume approach and element type. The code needs to be checked, verified and documented more completely.*

3.3.1.4 Assigning Material Properties

GWF domain material properties may vary on a cell-by-cell basis. MUT offers some simple instructions for selecting cells and assigning material property values to them. As you might imagine, this instruction:

choose all cells

Select all cells in the active model domain.

selects all cells in the *active* model domain.

This is an example of what we refer to as a *generic* instruction, which means it can be applied to any of the currently available model domains: GWF, SWF or CLN.

If our intention is to choose all cells in the GWF domain, we first need to activate it using this instruction:

active domain

1. **$_Domain** The name of the domain to be activated: GWF, SWF or CLN

This instruction activates the given domain named in **$_Domain** so that it will be used with generic instructions such as choose all cells.

So to activate the GWF domain, we would insert these instructions in the input file:

```
active domain
gwf
```

These instructions can be used to choose cells in various ways:

choose cell at xyz

1. **R_x1**, **R_y1**, **R_z1** An xyz coordinate triplet.

The cell closest to the given xyz coordinate triplet will be chosen.

choose cells by layer

1. **I_layer** The number of the layer to be chosen.

The cells in Modflow layer number **I_Layer** will be chosen. Remember that Modflow layers are one cell high and are numbered from the top to the bottom of the model domain.[9]

choose cells from file

1. **$_file** The file **$_file** containing a list of cell numbers.

The cells listed in the file **$_file** will be chosen. [10]

[9] See the verification example `MUT_Examples\1_VSF_Column` which uses the previous two instructions to define a constant head at the base and a recharge boundary condition at the top of the 1D column.

[10] See the verification example `MUT_Examples\1_Abdul_MODHMS` which uses this instruction to assign some cells as inactive.

choose cells from gb elements

1. **$_file** The GRID BUILDER chosen element file **$_file** containing information concerning the status, chosen or not chosen, of each element in the GRID BUILDER model domain.

If an element is flagged as chosen in the GRID BUILDER model domain then the corresponding cell will be chosen in the MODFLOW-USG^{Swf} model domain. [11]

choose cells from gb nodes

1. **$_file** The GRID BUILDER chosen node **$_file** containing information concerning the status, chosen or not chosen, of each node in the GRID BUILDER model domain.

If a node is flagged as chosen in the GRID BUILDER model domain then the corresponding cell will be chosen in the MODFLOW-USG^{Swf} model domain. [12]

The previous two instructions are used to choose cells for the mesh-centered and node-centered approaches respectively.

Cell selection instructions are cumulative. For example, you can modify the current selection by repeating instructions like choose cell at xyz or choose cells by layer with different inputs and then assign properties to the current selection. This instruction clears the selection before beginning new cell selection(s):

clear chosen cells

Clears the current cell selection.

It is good practice to clear the selection before starting a new selection. MUT echoes the results of the selection instructions to the screen and `.eco` file as shown in this example:

```
clear chosen cells
 GWF Cells chosen:          0

choose all cells
 GWF Cells chosen:      39765
```

If a cell selection instruction has unexpected results these are good places to check.

[11] See the verification example `MUT_Examples\1_Abdul_Prism_Cell` which uses this instruction to assign some cells as inactive.

[12] See the verification example `MUT_Examples\1_Abdul_Prism_Cell_nc` which uses this instruction to assign some cells as inactive.

These instructions can be used to assign material properties to the current cell selection:

gwf kh

1. **R_val** Horizontal hydraulic conductivity [$L\ T^{-1}$].

A horizontal hydraulic conductivity of **R_val** is assigned to the chosen cells.

gwf kv

1. **R_val** Vertical hydraulic conductivity [$L\ T^{-1}$].

A vertical hydraulic conductivity of **R_val** is assigned to the chosen cells.

gwf ss

1. **R_val** Specific storage [L^{-1}].

A specific storage of **R_val** is assigned to the chosen cells.

gwf sy

1. **R_val** Specific yield (-).

A specific yield of **R_val** is assigned to the chosen cells.

gwf alpha

1. **R_val** Van Genuchten/Brooks-Corey Alpha [L^{-1}].

A Van Genuchten/Brooks-Corey Alpha of **R_val** is assigned to the chosen cells.

gwf beta

1. **R_val** Specific yield (-).

A specific yield of **R_val** is assigned to the chosen cells.

gwf sr

1. **R_val** Residual saturation [-].

A residual saturation of **R_val** is assigned to the chosen cells.

gwf brooks

1. **R_val** Brooks-Corey exponent.

A Brooks-Corey exponent of **R_val** is assigned to the chosen cells.

Defining each different material property as described above can be tedious. A lookup table of GWF material properties is provided in the file `qryGWFMaterials.txt`, located in the `USERBIN` directory as outlined on page 7.

In order for MUT to access the lookup table, you first need to provide a link to this file using the instruction:

gwf materials database

1. **$_file** GWF material properties lookup table file name.

MUT uses the file **$_file** to look up GWF material properties.

You can now assign a full set of GWF material properties to the current cell selection, as described on page 30, using this instruction:

chosen cells use gwf material number

1. **I_val** GWF material ID number.

A unique set of GWF material properties is retrieved from a lookup table, using the given material ID number **I_val**, and assigned to the chosen cells.

You can find detailed information about how to use MICROSOFT ACCESS to modify or define your own lookup tables in Tutorial C.

During the model build, each GWF cell is assigned a zone number from either the layer number or 2D template zone number. Just like individual cells, zones can be selected using these instructions:

choose all zones

Select all zones in the active model domain.

choose zone number

1. **I_value** The number of the zone to be chosen.

clear chosen zones

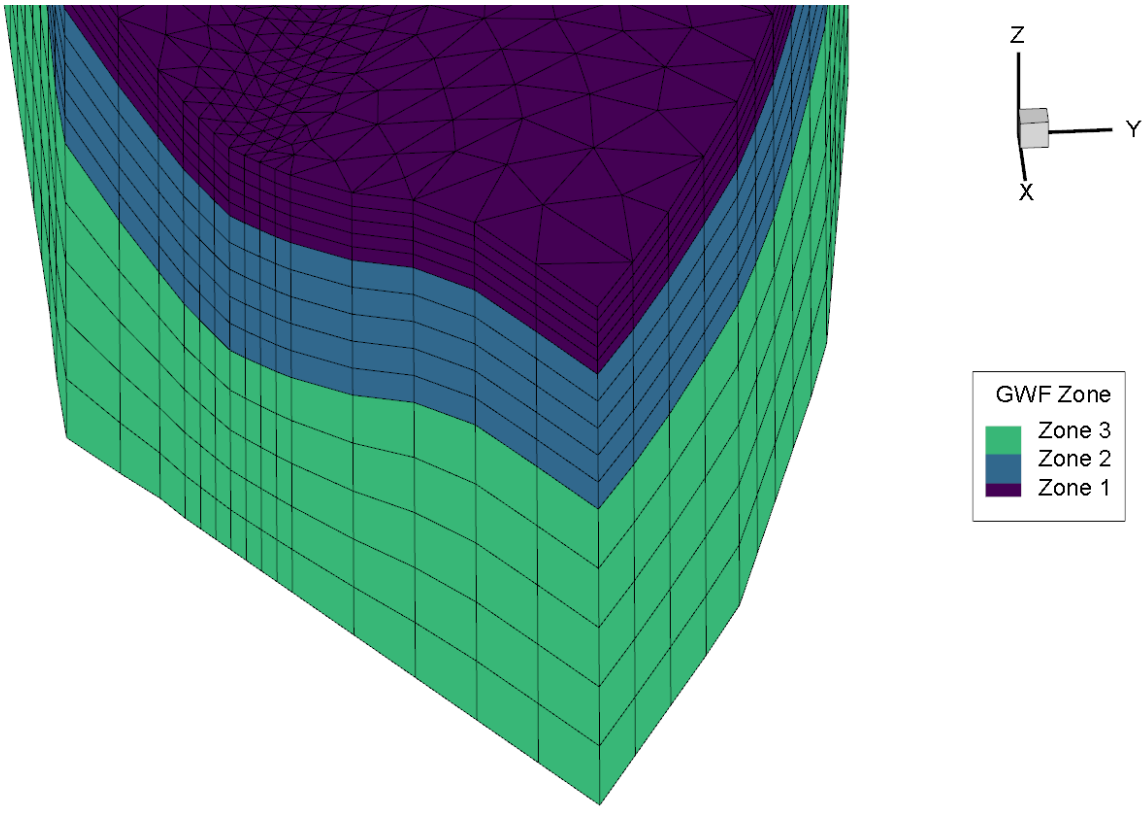
Clears the current zone selection.

The zone selection can be converted into a cell selection using this instruction:

choose cells by chosen zones

If a zone is currently chosen, any cell which has that zone number will be chosen. This cell selection can now be used to assign material properties.

Below is an example [13] of a case in which the element zone numbers have been assigned by layer number for a 3-layer case:



[13] See verification example `MUT_Examples\6_Abdul_Prism_Cell`

Some key features to note are:

- There are 3 zones, corresponding to the layers 1 to 3.
- 'Zone 1', coloured dark blue, corresponds to layer 1. Recall that the MODFLOW-USG^{Swf} mesh is generated from the top down, so layer 1 is at the top of the model domain.
- Each layer is composed of multiple MODFLOW layers, which are each one cell thick.

The following example uses the materials database and zone selection to assign material properties for the 3-layer case:

```
gwf materials database
 Materials file C:\MUT\MUT_Examples-main\_MUT_USERBIN\qryGWFMaterials.txt

active domain
 gwf

clear chosen zones
 GWF Zones chosen:          0

choose zone number
 Adding zone number:        1
 GWF zone numbers currently chosen:
     1

choose zone number
 Adding zone number:        2
 GWF zone numbers currently chosen:
     1
     2

choose zone number
 Adding zone number:        3
 GWF zone numbers currently chosen:
     1
     2
     3

clear chosen cells
 GWF Cells chosen:          0

choose cells by chosen zones
 GWF Cells chosen:      39765

chosen cells use gwf material number
 Assigning all chosen GWF cells properties of material              4, Borden sand
```

```
Kh_Kx:                  1.00000E-05
Kv_Kz:                  1.00000E-05
Specific Storage:       1.20000E-07
Specific Yield:         0.34000
Alpha:                   1.9000
Beta:                    6.0000
Sr:                     0.18000
Unsaturated Function Type:   Van Genuchten
```

Some key features to note are:

- As we choose zone numbers, the list of currently chosen zones grows accordingly.
- The final number of GWF cells chosen is equal to the total number of cells in the domain, since we had selected all 3 layers prior to converting the zone selection to a cell selection.
- After assignment, the GWF material number, name and assigned property values are echoed to screen and `.eco` file.

3.3.1.5 Initial Conditions

An initial (or starting) head should be assigned to each cell in the GWF domain. This could be an initial guess at the beginning of a transient stress period or a set of hydraulic heads from a previous run.

To assign a uniform hydraulic head to the GWF model domain, you must first make a cell selection as described on page 30, then use this instruction:

gwf initial head

1. **R_value** Initial (or starting) hydraulic head [L].

An initial hydraulic head of **R_value** is assigned to the chosen cells.

To assign a linearly varying head that is a function of z (i.e. depth or elevation), use this instruction:

gwf initial head function of z

1. **R_z(1)**, **R_head(1)** First $z, head$ pair.
2. **R_z(2)**, **R_head(2)** Second $z, head$ pair.
3. **...**
4. **R_z(n)**, **R_head(n)** nth $z, head$ pair.

An initial head is calculated for each chosen cell, based on it's z-coordinate location, by interpolating a head from the given list of $z, head$ pairs.

This subtask reads a list of $z, head$ pairs until an end instruction is encountered e.g.

end gwf initial head function of z

This is commonly used to generate an initial head for a simple column model [14] as shown here: :

```
gwf initial head function of z
!  z    head
  0.0  -100.0
100.0     0.0
```

Some key features of this example are:

- The two given $z, head$ pairs define an initial head that varies from $head = -100.0$ at $z = 0$ to $head = 0.0$ at $z = 100.0$. You may supply as many pairs as needed to define the initial head.
- z-coordinates must increase continuously from the top of the list to the bottom.
- the z-range of the supplied pairs should cover the entire z-range of the model domain.
- For each node in the model domain mesh, the z coordinate is used to interpolate an initial head (i.e. *head* value) using the appropriate $z, head$ pair.

3.3.1.6 Boundary Conditions

A constant head boundary condition fixes the head at a GWF cell at a given value, allowing water to flow into or out of the GWF model domain depending on surrounding conditions. To assign a uniform constant head to the GWF model domain use this instruction:

gwf constant head

1. **R_value** Constant hydraulic head [L].

An constant hydraulic head of **R_value** is assigned to the chosen cells.

A drain boundary condition allows water to flow out of the GWF model domain if the hydraulic head of the cell is higher than the drain elevation. To add a drain to the GWF model domain use this instruction:

gwf drain

1. **R_value** Drain conductance [L/T].

A drain conductance of **R_value** is assigned to the chosen cells. The top elevation of the cell is assigned automatically as the drain elevation

[14] The verification example `MUT_Examples\1_VSF_Column` uses the `gwf initial head function of z` instruction to define the initial head of the model domain


Should we add an instruction where the drain elevation is specified?

A recharge boundary condition forces water to flow in to the GWF model domain at a specified rate. To add a recharge to the GWF model domain use this instruction:

gwf recharge

1. **R_value** Recharge rate [L/T].
2. **I_option** Recharge option.

A recharge rate of **R_value** is assigned to the chosen cells.

The recharge option **I_option**is used to define where the recharge is to be applied and can have one of the following values:

1. To top layer
2. To one specified node in each vertical column
3. To highest active node in each vertical column
4. To the swf domain on top of each vertical column


Sould we have an example of pumping with an assigned nodal flux? Should it be from a CLN? Is there a boundary condition that assures the head will not be drawn down below the pumping node elevation?

3.4 Surface Water Flow(SWF) Domain

The SWF domain is a 2D network of cells which is usually, but not necessarily, coincident with the top of the GWF domain.

MODFLOW-USG^{Swf} allows individual (i.e. GWF, SWF and CLN) processes to add to the global conductance matrix in order to represent fluxes between cells within a process as well as with cells of other processes. MODFLOW-USG^{Swf} provides a framework for tightly coupling multiple hydrologic processes. The tight coupling, in contrast to a sequential or iterative coupling approach, occurs through the formulation of a global conductance matrix that includes the cells for all processes.

The flows between SWF cells are governed by the diffusion-wave equations, which ultimately provide a pressure head (i.e. surface water depth) at each cell.

3.4.1 Generating a SWF Domain

A SWF domain can be easily added to the MODFLOW-USG^{Swf} model using the same template mesh that was used to define the GWF mesh, as described in Section 3.2.

The SWF domain is generated using this instruction:

> **generate swf domain**
> This subtask currently has only one instruction that is used to define:
>
> - Top elevation (i.e. z-coordinate, ground surface elevation)
>
> It reads instructions until an end instruction is found e.g.
>
> **end generate swf domain**

Here, the top elevation instruction has the same options as described in Section 3.3.1.1 for the GWF domain.

Currently, the element zone numbering for the SWF domain is determined by the instruction used to generate the template mesh:

2d mesh from gb The GRID BUILDER element area numbers are used to define the MODFLOW-USG^{Swf} element zone numbers.

generate uniform rectangles The element zone numbers default to 1.


We should provide instructions for defining swf zone numbers, e.g. from a cell list or raster file.

3.4.1.1 Cell Connection Properties


SWF cell to SWF cell connection properties are automatically defined by MUT *based on control-volume approach and element type. The code needs to be checked, verified and documented more completely.*

These SWF cell connection properties vary on a cell-by-cell basis:

- The SWF-GWF connection length.
- The SWF-GWF connection area.

Cell selections must first be made using the instructions described on page 30 for the GWF domain.

Currently, MUT assigns a default value of 0.001 m for the SWF cell to GWF cell connection length, and this instruction allows you to change it:

swf to gwf connection length

1. **R_value** SWF cell to GWF cell connection length $[L]$.

A SWF cell to GWF cell connection length **R_value** is assigned to the chosen cells.

Currently, MUT assigns a SWF-GWF connection area depending on which control volume approach is used :

Mesh-centred approach The connection area will be calculated as the template mesh element area (i.e. the yellow triangle shown on page 22).

Node-centred approach The connection area will be calculated as the contibuting area of neighbouring template mesh elements (i.e. the yellow polygon shown on page 23).

3.4.1.2 Assigning Material Properties

Unlike the GWF domain, SWF domain material properties vary on a zone-by-zone basis, which means assigning material property values are done using zone selections instead of cell selections.

Prior to making cell or zone selections and assigning properties, we need to activate the SWF domain using these instructions:

```
active domain
swf
```

Zone selections must first be made using the instructions described on page 34 for the GWF domain, then these instructions can be used to assign material properties to the current zone selection:

swf manning

1. **R_value** Manning's coefficient of friction $[L^{-1/3}T]$.

A Manning's coefficient of **R_value** is assigned to the chosen cells.

swf depression storage height

1. **R_value** Depression storage height $[L]$.

A depression storage height of **R_value** is assigned to the chosen zones.

swf obstruction storage height

1. **R_value** Obstruction storage height [L].

An obstruction storage height of **R_value** is assigned to the chosen zones.

swf depth for smoothing

1. **R_value1** Depth for smoothing height 1 [L].
2. **R_value2** Depth for smoothing height 2 [L].

Two depth for smoothing heights are read in **R_value1** and **R_value2** and assigned to the chosen zones.

A lookup table of SWF material properties is provided in the file `qrySWFMaterials.txt`, located in the `USERBIN` directory as outlined on page 7.

In order for MUT to access the lookup table, you first need to provide a link to this file using the instruction:

swf materials database

1. **$_file** SWF material properties lookup table file name.

MUT uses the file **$_file** to look up SWF material properties.

You can now assign a full set of SWF material properties to the current zone selection, as described on page 34, using this instruction:

chosen zones use swf material number

1. **I_val** SWF material ID number.

A unique set of SWF material properties is retrieved from a lookup table, using the given material ID number **I_val**, and assigned to the chosen zones.

The assigned SWF material properties are written to the screen and `.eco` file:

```
chosen zones use swf material number
 Assigning all chosen SWF zones properties of material 1.0000, Streambed
 Manning's Coefficient:          3.00000E-02
 Depression Storage Height:      0.10000
 Obstruction Storage Height:      0.0000
 SWF Smoothing Depth 1:          1.00000E-06
 SWF Smoothing Depth 2:          1.00000E-06
```

You can find detailed information about how to use Microsoft Access to modify or define your own lookup tables in Tutorial C.

3.4.1.3 Initial Conditions

An initial (or starting) head should be assigned to each cell in the SWF domain. This could be an initial guess at the beginning of a transient stress period or a set of hydraulic heads from a previous run.

To assign an initial head to the SWF model domain, you must first make a cell selection as described on page 30, then this instruction can be used to calculate an initial (or starting) head for the flow solution given an initial surface water depth:

swf initial depth

1. **R_value** Initial depth [L].

An initial depth of **R_value** is used to calculate an initial head at each of the chosen cells.

3.4.1.4 Boundary Conditions

A constant head boundary condition fixes the head at a SWF cell at a given value, allowing water to flow into or out of the SWF model domain depending on surrounding conditions. To assign a uniform constant head to the SWF model domain use this instruction:

swf constant head

1. **R_value** Constant hydraulic head [L].

An constant hydraulic head of **R_value** is assigned to the chosen cells.

A recharge boundary condition forces water to flow in to the SWF model domain at a specified rate. To add recharge to the SWF model domain use this instruction:

swf recharge

1. **R_value** Recharge rate [L/T].
2. **I_option** Recharge option.

A recharge rate of **R_value** is assigned to the chosen cells.

The recharge option **I_option** is used to define where the recharge is to be applied and in this case should be set to a value of 4, which applies the recharge to the SWF domain.

A critical depth boundary condition assigned to a SWF cell allows water to flow out of the SWF model domain at a rate that depends on the surface water depth and a contributing length (i.e. representing the length of the cell side over which the outflow occurs).

Cell selections must first be made then one of the following two instructions can be used to assign a critical depth outflow boundary condition to the SWF model domain:

swf critical depth with sidelength1

A critical depth outflow boundary condition is assigned to the chosen cells.

It is assumed that an accurate estimate of the contributing length of a cell can be based on a template element side length. In this case we have arbitrarily used side 1.

swf critical depth

A critical depth outflow boundary condition is assigned to the chosen cells.

The contributing length of the cell outflow boundary is calculated from the SWF mesh outer boundary nodes, with each outer boundary node connected to a chosen cell contributing a half-element side length in both directions along the outer boundary.

Although swf critical depth is less convenient than swf critical depth with sidelength1, it does calculate a contributing length that matches the actual length along the SWF mesh outer boundary.

The verification example `MUT_Examples\6_Abdul_Prism_Cell` uses the second approach to define the critical depth outflow boundary condition:

```
clear chosen nodes
choose gb nodes
./gb/grid.nchos.Outer boundary nodes
flag chosen nodes as outer boundary

clear chosen cells
clear chosen nodes
choose cells from gb elements
./gb/grid.echos.Critical depth outlet
swf critical depth
```

Some key features of this example are:

- SWF *nodes* are chosen and flagged to be on the outer boundary with the instruction flag chosen nodes as outer boundary.
- SWF *cells* are chosen using a GRID BUILDER chosen *elements* file with the instruction choose cells from gb elements. Because this example was generated using the mesh-centred control volume approach, there is a 1-to-1 correspondence between template mesh elements and SWF cells.

In the example `MUT_Examples\6_Abdul_Prism_Cell_nc`, the node-centred control volume approach is used and the instruction choose cells from gb nodes is used instead, because in this case there is a 1-to-1 correspondence between template mesh *nodes* and SWF cells.

The SWF and GWF meshes that MUT generates inherit node and element information from the template mesh. Currently, you can define node selections using these instructions:

choose all nodes

Select all nodes in the active model domain.

choose node at xyz

1. **R_x1**, **R_y1**, **R_z1** An xyz coordinate triplet.

The node closest to the given xyz coordinate triplet will be chosen.

choose gb nodes

1. **$_file** The GRID BUILDER chosen node **$_file** containing information concerning the status, chosen or not chosen, of each node in the GRID BUILDER model domain.

If a node is flagged as chosen in the GRID BUILDER model domain then the corresponding node will be chosen in the MODFLOW-USG^{Swf} model domain.

clear chosen nodes

Clears the current node selection.

3.5 Connected Linear Network(CLN) Domain

A CLN domain is a quasi-3D network of modflow cells which are each defined by individual line segments. The flows between CLN cells are governed by either open- or closed-channel flow equations, depending on surrounding conditions, which ultimately provide a pressure head or water depth at each cell.

MUT adds the CLN process equations to the global conductance matrix in order to represent fluxes between cells within the process as well as with cells of other processes.

The current version of MUT has the following limitations for the definition of the CLN domain:

1. General CLN domains, where the cell geometry is independent of the GWF mesh and cell-to-cell connection can be one-to-many or many-to-one are not yet implemented. MUT assumes a 1-to-1 connection exists between CLN cells and GWF layer 1 cells (i.e. top layer).

2. CLN flows to the SWF domain are not yet implemented, just CLN flows to the GWF domain.

3. No CLN domain boundary conditions have been implemented.

This is sufficient for the short-term purpose of solving the verification example `MUT_Examples\3_1_CLN_for_SWF`, which compares the use of a CLN versus an SWF domain for simulating flow in a surface water domain coupled to a GWF domain, but not for solving more general problems of interest.

3.5.1 Generating a CLN Domain

A CLN domain can be added to the MODFLOW-USG^{Swf} model using this instruction:

---

generate cln domain

This subtask is currently limited to defining the CLN domain from a single pair of xyz coordinates and a specified number of cells.

It reads instructions until an end instruction is found e.g.

end generate cln domain

This instruction can be used to define a simple CLN domain:

---

cln from xyz pair

1. **R_x1**, **R_y1**, **R_z1** First xyz coordinate triplet.
2. **R_x2**, **R_y2**, **R_z2** Second xyz coordinate triplet.
3. **I_nCells** Number of cells in the CLN.

The 2 given xyz coordinates define the endpoints of a line defining the CLN. The CLN is subdivided into **I_nCells** individual cells.

In the verification example `MUT_Examples\3_1_CLN_for_SWF`, the inputs are defined so the CLN cells coincide exactly with the top layer of GWF cells as shown below:

```
generate uniform rectangles
101.0, 101   !  Mesh length in X-direction and number of rectangular elements
1.0, 1   !  Mesh length in Y-direction and number of rectangular elements


generate layered gwf domain

    top elevation
        elevation from xz pairs
               0.0, 2.0
             101.0, 1.0
```

```
        end elevation from xz pairs
    end top elevation

    ...

 generate cln domain
    cln from xyz pair
          0.0    0.5     2.0
        101.0    0.5     1.0
        101  ! number of new CLN cells

end generate cln domain
```

Some key features of this example are:

- A template mesh of length 101.0 and with 101 elements is used to define the GWF domain.
- The top of the GWF domain slopes from $z = 2.0$ at $x = 0.0$ to $z = 1.0$ at $x = 101.0$.
- Because CLN domains are not necessarily dependent on GWF meshes, it does not use the template mesh, but instead generates a CLN domain using the `cln from xyz pair` instruction. The instruction is set up to generate 101 CLN cells that also slope from $z = 2.0$ at $x = 0.0$ to $z = 1.0$ at $x = 101.0$

3.5.1.1 Assigning Material Properties

CLN domain material properties vary on a zone-by-zone basis. In the current version of MUT the assignment of CLN material properties is very rudimentary.

Prior to assigning properties, we need to activate the CLN domain using these instructions:

```
active domain
cln
```

A lookup table of CLN material properties is provided in the file `qryCLNMaterials.txt`, located in the `USERBIN` directory as outlined on page 7.

In order for MUT to access the lookup table, you first need to provide a link to this file using the instruction:

cln materials database

1. **$_file** CLN material properties lookup table file name.

MUT uses the file **$_file** to look up CLN material properties.

Zone selections must first be made using the instructions described on page 34 for the GWF domain.

You can now assign a full set of CLN material properties to the current zone selection, as described on page 34, using this instruction:

chosen zones use cln material number

1. **I_val** CLN material ID number.

A unique set of CLN material properties is retrieved from a lookup table, using the given material ID number **I_val**, and assigned to the chosen zones.

The assigned CLN material properties are written to the screen and `.eco` file:

```
chosen zones use cln material number
 Assigning all chosen CLN zones properties of material              3, CLN_for_SWF
 Geometry:           Rectangular
 Rectangular Width:       1.0000
 Rectangular Height:      1.0000
 Direction:          Horizontal
 Flow Treatment:     Unconfined/Mannings
 Longitudinal K:         5.48000E-02
```

You can find detailed information about how to use Microsoft Access to modify or define your own lookup tables in Tutorial C.

3.5.1.2 Initial Conditions

An initial (or starting) head should be assigned to each cell in the CLN domain. This could be an initial guess at the beginning of a transient stress period or a set of hydraulic heads from a previous run.

To assign an initial head to the CLN model domain, you must first make a cell selection as described on page 30, then this instruction can be used to calculate an initial (or starting) head for the flow solution given an initial water depth:

cln initial depth

1. **R_value** Initial depth [L].

An initial depth of **R_value** is used to calculate an initial head at each of the chosen cells.

3.5.1.3 Boundary Conditions

No CLN domain boundary conditions have been implemented in the current version of MUT.

In the verification example `MUT_Examples\3_1_CLN_for_SWF`, all boundary conditions are applied to the GWF domain.

3.6 Stress Periods

A MODFLOW-USG^{Swf} simulation can be broken up into separate periods of time called "stress periods". Boundary conditions can be defined at the beginning of each stress period and changed in subsequent stress periods.

At least one stress period must be defined using this instruction:

stress period

This subtask has several instructions that can be used to define the duration, type and timestepping parameters of the stress period.

It reads instructions until an `end` instruction is found e.g.

end stress period

These instructions can be used to define the stress period parameters:

type

1. **$_type** Stress period type.

The stress period type is defined by the string **$_type**. It can be one of the following:

- SS A steady-state stress period in which the simulation is carried out until it reaches a state of equilibrium with the defined boundary conditions.
- TR A transient stress period in which the simulation is carried out for a specified duration with the defined boundary conditions.

duration

1. **R_value** Stress period duration [T].

The stress period duration, is defined by the string **R_value**. It should be entered using the correct units of time as outlined in Section 3.1.

number of timesteps

1. **I_value** Number of timesteps to be used for this stress period.

You can change the default starting time step size of 1×10^{-3} seconds with this instruction:

deltat

1. **R_value** Starting time step size [T].

The starting time step size used for the stress period is defined by the string **R_value**. It should be entered using the correct units of time as outlined in Section 3.1.

You can change the default minimum time step size of 1×10^{-5} seconds with this instruction:

tminat

1. **R_value** Minimum time step size [T].

The minimum time step size to allow for the stress period is defined by the string **R_value**. It should be entered using the correct units of time as outlined in Section 3.1.

You can change the default maximum time step size of 60.0 seconds with this instruction:

tmaxat

1. **R_value** Maximum time step size [T].

The maximum time step size to allow for the stress period is defined by the string **R_value**. It should be entered using the correct units of time as outlined in Section 3.1.

You can change the default multiplier for time step size of 1.1 with this instruction:

tadjat

1. **R_value** Multiplier for time step size [T].

The multiplier for adjusting time step size when using adaptive time-stepping is defined by the string **R_value**.

You can change the default divider for time step size of 2.0 with this instruction:

tcutat

1. **R_value** Divider for time step size [T].

The divider for adjusting time step size when using adaptive time-stepping is defined by the string **R_value**.

To add more stress periods, repeat the stress period subtask instructions and boundary condition definitions as many times as required. Stress periods are numbered automatically as they are added.

Here is an example which could be used to define two stress periods:

```
! stress period 1
stress period
    type
    TR

    duration
    3000.0d0
end stress period

active domain
swf
    choose all cells
    swf recharge
    5.56d-6
    4

    clear chosen nodes
    choose cell at xyz
    0.0 0.0 0.0
    swf constant head
    1.0

! stress period 2
stress period
    type
    TR

    duration
    3000.0d0
end stress period

active domain
swf
    choose all cells
    swf recharge
    0.0d0
    4
```

Some key features of this example are:

- Both stress periods are transient (type TR) with a duration of 3000.
- The recharge applied to the SWF domain (recharge option 4) is 5.5e-6 for the first stress period, then is reduced to 0.0 in the second stress period.
- The constant head applied to the SWF domain in stress period 1 is maintained for the entire simulation. By default, a boundary condition is maintained through subsequent stress periods unless it is redefined.
- Any boundary conditions given after an end stress period instruction apply to that stress period until another stress period instruction is encountered.

3.7 Output Control

This instruction can be used to generate a MODFLOW-USG^{Swf} output control file:

generate output control file

1. **R_t(1)** First output time [T].
2. **...**
3. **R_t(n)** nth output time [T].

This subtask reads a list of output times **R_t()** until an end instruction is encountered e.g.

end generate output control file

The verification example `MUT_Examples\1_Abdul_prism_cell` generates an output control file with 10 output times using these instructions:

```
! ----------------------------------Output Control
generate output control file
    1e-4
    60.
    300.0
    600.0
    900.0
    1200.0
    1500.0
    3000.0
    4500.0
    6000.0
end generate output control file
```

The output control file looks like this:

```
# MODFLOW-USG OC file written by Modflow-User-Tools version 1.28
ATSA NPTIMES   10
  9.999999747378752E-005   60.0000000000000        300.000000000000
   600.000000000000        900.000000000000        1200.00000000000
   1500.00000000000        3000.00000000000        4500.00000000000
   6000.00000000000
HEAD SAVE UNIT   114
HEAD PRINT FORMAT 0
DRAWDOWN SAVE UNIT   115
DRAWDOWN PRINT FORMAT 0
PERIOD     1
    DELTAT   1.0000E-03
    TMINAT   1.0000E-05
    TMAXAT    60.00
    TADJAT    1.100
    TCUTAT    2.000
        SAVE HEAD
        PRINT HEAD
        SAVE DRAWDOWN
        SAVE BUDGET
        PRINT BUDGET
PERIOD     2
    DELTAT   1.0000E-03
    TMINAT   1.0000E-05
    TMAXAT    60.00
    TADJAT    1.100
    TCUTAT    2.000
        SAVE HEAD
        PRINT HEAD
        SAVE DRAWDOWN
        SAVE BUDGET
        PRINT BUDGET
```

Some key features of this example are:

- MUT automatically inserts the adaptive time-stepping option (`ATSA`) in the file, defines the number of print times in the simulation (NPTIMES 10) and the list of print (i.e. output) times.
- Two stress periods were defined and the listed parameters are using the default values.

3.8 Solver Parameters

A lookup table of MODFLOW-USG^{Swf} solver parameters is provided in the file `qrySMS.txt`, located in the `USERBIN` directory as outlined on page 7.

In order for MUT to access the lookup table, you first need to provide a link to this file using the instruction:

sms database

1. **$_file** Solver parameters lookup table file name.

MUT uses the file **$_file** to look up the solver parameter values.

You can now assign the full set of solver parameters using this instruction:

sms parameter set number

1. **I_val** Solver parameter set ID number.

A unique set of solver parameter values is retrieved from a lookup table, using the given parameter set ID number **I_val**.

You can find detailed information about how to use MICROSOFT ACCESS to modify or define your own lookup tables in Tutorial C.

Currently, all of the verification examples use default solver parameters, which are defined in the input file as shown in this example:

```
sms database
qrySMS.txt
sms parameter set number
1
```

The solver parameter values are written to the screen and `.eco` file:

```
sms parameter set number
 Using SMS parameter set    1, Default
 OUTER ITERATION CONVERGENCE CRITERION  (HCLOSE)           1.00000E-03
 INNER ITERATION CONVERGENCE CRITERION  (HICLOSE)          1.00000E-05
 MAXIMUM NUMBER OF OUTER ITERATIONS     (MXITER)                  250
 MAXIMUM NUMBER OF INNER ITERATIONS     (ITER1)                   600
 SOLVER PRINTOUT INDEX                  (IPRSMS)                    1
 NONLINEAR ITERATION METHOD             (NONLINMETH)                1
 LINEAR SOLUTION METHOD                 (LINMETH)                   1
 D-B-D WEIGHT REDUCTION FACTOR          (THETA)             0.70000
 D-B-D WEIGHT INCREASE INCREMENT        (KAPPA)             0.10000
 D-B-D PREVIOUS HISTORY FACTOR          (GAMMA)             0.10000
 MOMENTUM TERM                          (AMOMENTUM)          0.0000
 MAXIMUM NUMBER OF BACKTRACKS           (NUMTRACK)                200
 BACKTRACKING TOLERANCE FACTOR          (BTOL)               1.0000
```

BACKTRACKING REDUCTION FACTOR	(BREDUC)	0.20000	
BACKTRACKING RESIDUAL LIMIT	(RES_LIM)	1.0000	
TRUNCATED NEWTON FLAG	(ITRUNCNEWTON)	0	
Options		SOLVEACTIVE	DAMPBOT
ACCELERATION METHOD	(IACL)	1	
EQUATION ORDERING FLAG	(NORDER)	0	
LEVEL OF FILL	(LEVEL)	7	
MAXIMUM NUMBER OF ORTHOGONALIZATIONS	(NORTH)	14	
INDEX FOR USING REDUCED SYSTEM	(IREDSYS)	0	
RESIDUAL REDUCTION CONVERGE CRITERION	(RRCTOL)	0.0000	
INDEX FOR USING DROP TOLERANCE	(IDROPTOL)	1	
DROP TOLERANCE VALUE	(EPSRN)	1.00000E-03	

Chapter 4

Modflow-Usg^{Swf} Execution and Post-Processing

The steps in the model execution workflow are:

1. Run Modflow to create the new project output files (e.g. time-varying hydraulic head, drawdown etc).
2. Run `_post.mut` to post-process the Modflow project, which produces TECPLOT output files for the various Modflow domains (i.e. GWF,SWF and/or CLN) created during the Modflow simulation.
3. Run TECPLOT and examine the Modflow output files.

In this tutorial, we will build a 3D fully-coupled GWF-SWF model, check the build using TECPLOT, run MODFLOW-USG^{Swf} to generate output, then examine the results using TECPLOT.

```
! This example builds a modflow project of the Abdul Field Experiment
! The\swf\ mesh and top of the\gwf\ mesh are defined by a 2D Grid Builder triangular mesh
build modflow usg
```

Any input line beginning with is considered to be a comment and will be ignored by MUT!Here we begin the file with two comments describing the project.

The third line activates the MUT 'build modflow usg' environment, which accepts further instructions required to define the project. This environment can be split into roughly 4 sections:

Grid definition Instructions for defining the GWF,SWF andCLN numerical discretizations.

Modflow parameters Instructions for supplying Modflow parameter values (e.g. solver inputs for the SMS package, hydraulic properties for the LPF package etc.)

Stress periods, boundary conditions These instructions are repeated once for each desired stress period and include instructions about time stepping parameters and boundary conditions that are to be applied.

Output control Instructions defining a list of output times at which Modflow output files (e.g. heads, drawdowns, cell-by-cell flows etc.) are to be written.

The first group of instructions are used to build the Modflow unstructured mesh. MUT requires a 2D 'template'

```
! ------------------------------------Grid definition
2d mesh from gb
./gb/grid
```

Step 1: Copy an Existing Mut Project

Create a new folder called e.g. `My_Project` and copy the folder `MUT_Examples\6_Abdul_Prism_Cell` into it. Figure 4.1 shows the contents of our `6_Abdul_Prism_Cell` folder. Yours may look different depending on the root drive and folder location.

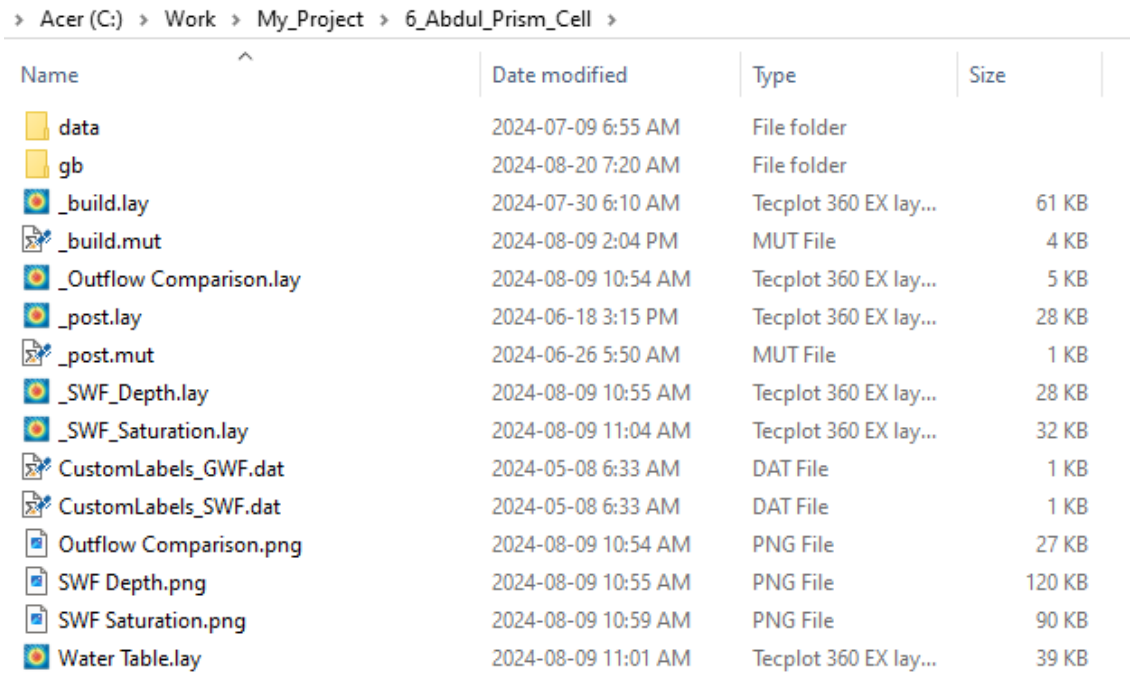


Figure 4.1: The contents of the 6_Abdul_Prism_Cell folder

This example contains several files you might typically find in a MUT Modflow project, including MUT input files (extension .mut), TECPLOT layout files (extension .lay), TECPLOT input files (extension .dat). For now, our focus will be on the MUT input file `_build.mut`.

Step 2: Modify the Input File(s)

The input file `_build.mut` is set up to build a Modflow project. If you open the file in a text editor you will see that it consists of a sequence of comments (lines beginning with an exclamation mark `!`), MUT instructions and data (numbers or alphanumeric strings). Details of the input file contents are described in detail in Section **??**. For now, we will only make a minor change to the input file before moving on to the next step, which is to add a new comment line of your choice at the start of the file.

Step 3: Execute Mut to Build the Project

Assuming you have followed the set-up instructions in Section **??**, you can execute MUT with the input file `_build.mut` by typing:

```
mut _build
```

Step 1: Run Modflow to Generate Output

Step 2: Run Mut to Post-Process the Modflow Output

Step 3: Run Tecplot to Visualize the Modflow Output

Chapter 5

Model Verification

5.1 1D Variably-saturated Flow in a Column

This example [1] simulates variably-saturated flow in a 1D column of homogeneous sand 100 m thick. Parameter values used in this example are shown in Table 5.1.

The Van Genuchten unsaturated function type was used.

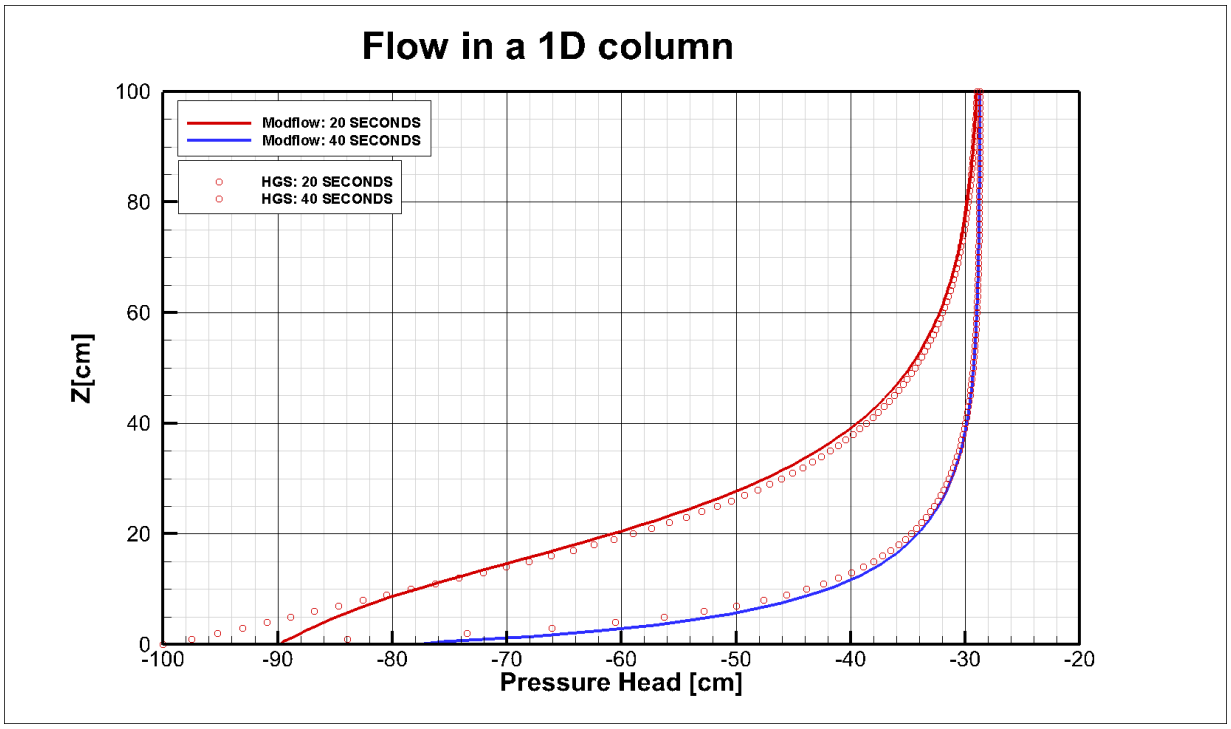
A uniform rainfall of 0.4 m/s was applied to the top of the column and the base was fixed at a pressure head of -100 m.

[1] See verification example `MUT_Examples\1_VSF_Column`

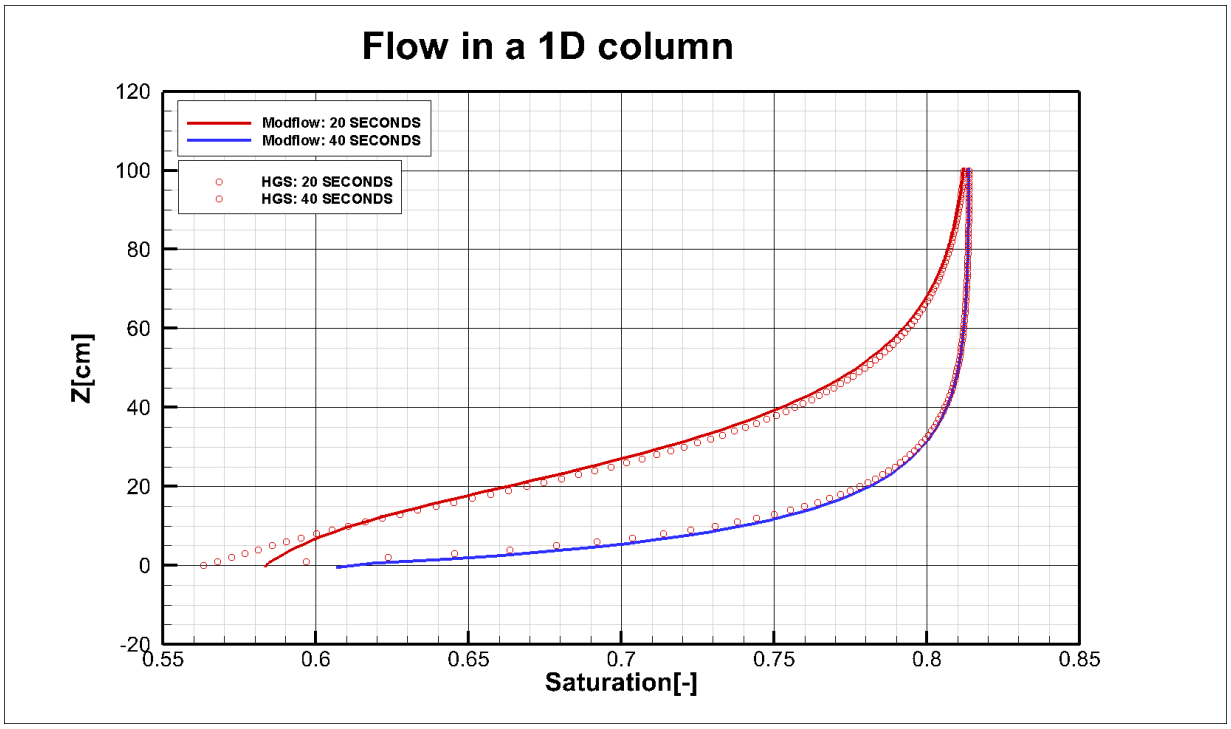
Table 5.1: Parameter Values for Simulation of the 1-D Column.

Parameter	Value	Unit
specific yield (porosity)	0.34	
hydraulic conductivity	1×10^{-5}	m s^{-1}
specific storage coefficient	1.2×10^{-7}	m^{-1}
Van Genuchten parameter	1.9	m^{-1}
Van Genuchten parameter	6	
residual saturation	0.18	
Manning coefficient for plot	0.3	s $m^{-1/3}$
Manning coefficient for channel	0.03	s $m^{-1/3}$
Initial water table elevation	2.78	m

Here is a comparison of pressure head versus depth results for MODFLOW-USG^{Swf} and HYDROGEOSPHERE at 20 and 40 seconds:



Here is a comparison of saturation versus depth results for MODFLOW-USG^{Swf} and HYDROGEOSPHERE at 20 and 40 seconds:



Although these results look good, there is a discrepancy between the unit systems used in the two models (e.g. K is 10 in HydroGeoSphere *and 1e-5 in* Modflow-Usg^{Swf}*. HGS column height appears to be 100 cm, while* Modflow-Usg^{Swf} *model is 100 m thick.* Mut *needs to be modified to allow the user to change unit systems and then we should update this example.*

5.2 2D Variably-saturated Flow in a Hillslope: Drains vs Surfacewater Flow

This example [2] simulates variably-saturated flow in a 2D hillslope of homogeneous material which receives a uniform recharge rate of 1.27 x 10^{-8} m/s was applied directly to the GWF domain at the ground surface. .

These are the properties we used:

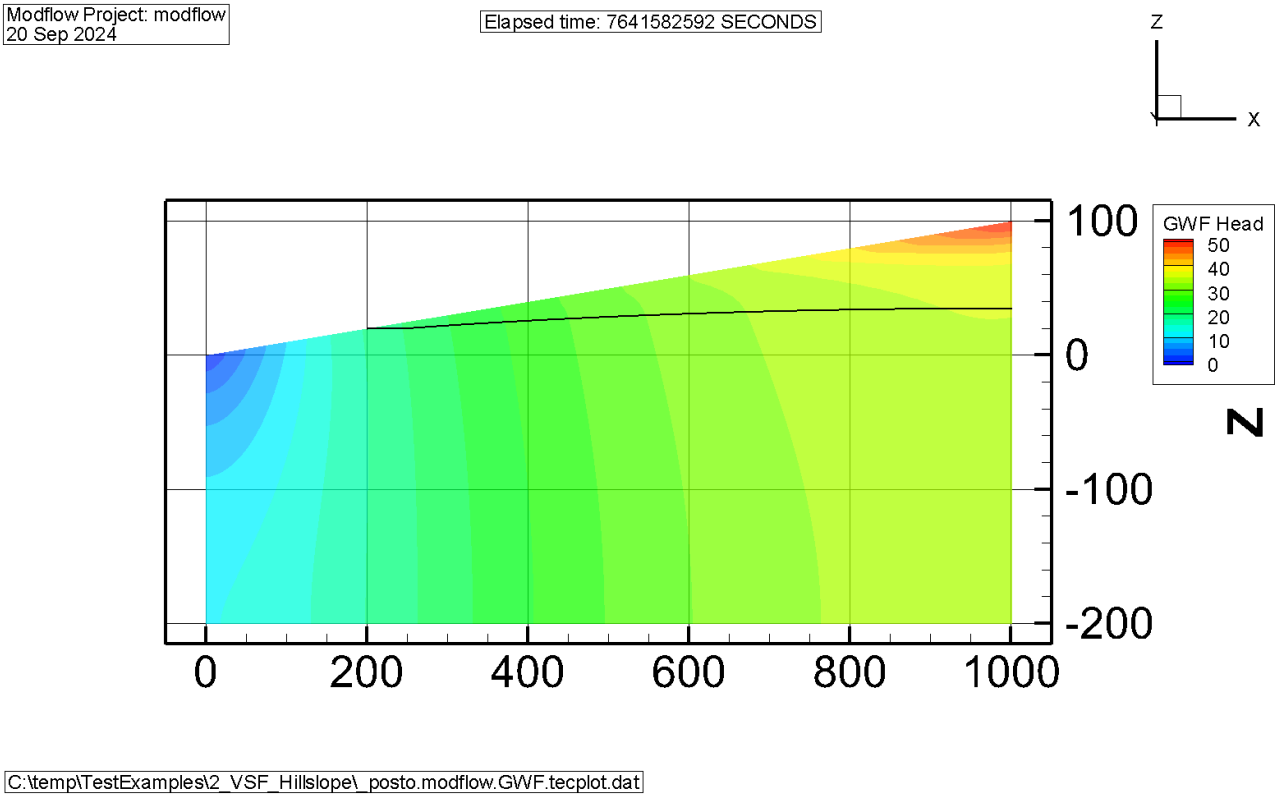
Variable name	Value	Units
Kh_Kx:	1.0E-06	m/s
Kv_Kz:	1.0E-06	m/s
Specific Storage:	1.0E-05	1/m
Specific Yield:	0.1	-
Alpha:	3.34E-02	1/m
Beta:	1.982	1/m
Sr:	0.2771	-

The Van Genuchten unsaturated function type was used.

A drain boundary condition was with a drain conductance of 1000 m/s was applied to the GWF domain at the ground surface.

Here is the hydraulic head distribution at 242 years (equilibrium):

[2] See verification example `MUT_Examples\2_VSF_Hillslope`



Some key features of this example are:

- The hill slopes from an elevation of 0 m at $x = 0$ to 1000 m at $x = 1000$.
- The base is flat at an elevation of -200 m.
- The water table is shown as heavy black line.

5.3 3D Fully-coupled Groundwater-Surface Water Flow: Abdul's Experiment

Properties from the HGS manual are shown in Table 5.2.

Note: the porosity is different than our value for specific yield above.

Table 5.2: Parameter Values for Simulation of the 3-D Field Scale Study of *Abdul* [1985].

Parameter	Value	Unit
porosity, Θ	0.37	
hydraulic conductivity, K	1×10^{-5}	m s^{-1}
storage coefficient, S_s,	1.2×10^{-7}	m^{-1}
Van Genuchten parameter, α	1.9	m^{-1}
Van Genuchten parameter, β	6	
residual saturation, S_r	0.18	
Brooks-Corey coefficient, n	3.4	
Manning coefficient for plot	0.3	s $m^{-1/3}$
Manning coefficient for channel	0.03	s $m^{-1/3}$
Initial water table elevation	2.78	m

Chapter 6

Illustrative Example

We intend to present a watershed scale example of a fully-coupled GWFSWF system.

Appendix A

GitHub Useage in Microsoft Visual Studio

Appendix B

Tecplot Useage

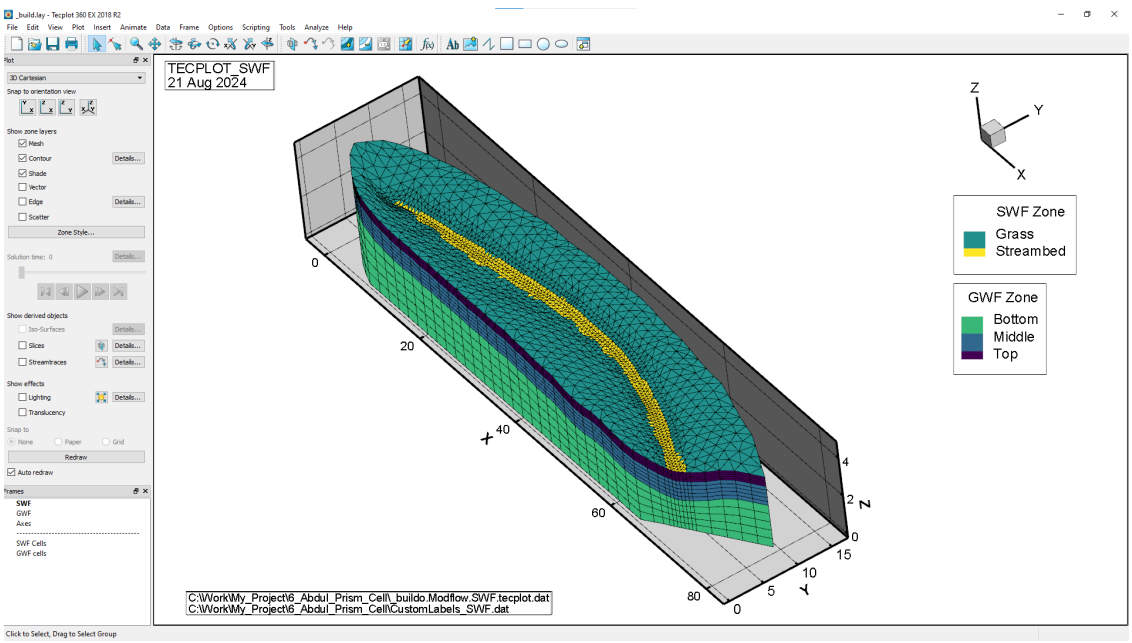
Step 4: Run Tecplot to Examine the Built Project

You can run Tecplot and load the Tecplot layout file `_build.lay` by typing:

```
tec360 _build.lay
```

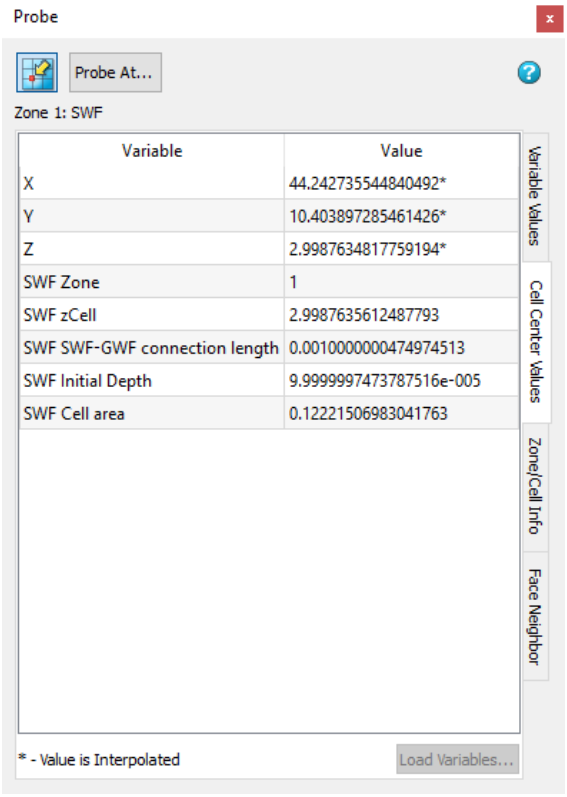
Some key features to note are:

- A Tecplot window should open:



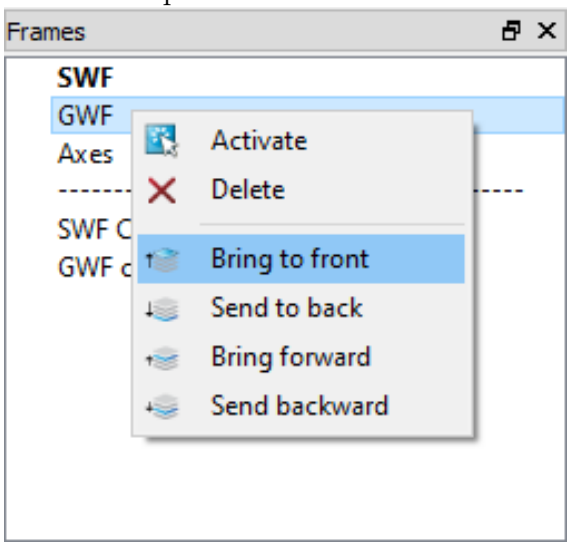
This Tecplot layout file has been constructed with multiple frames (see lower left 'Frames' window) showing details about theSWF andGWF model domains. This default view shows the distribution of the various materials defined in the model, such as theSWF domain materials called 'Grass' and 'Streambed'. Detailed information about manipulating the data in Tecplot to produce the desired plots is discussed in Section **??**.

- TECPLOT data can be probed using the probe tool  . Here we see the results of probing a location in theSWF domain:

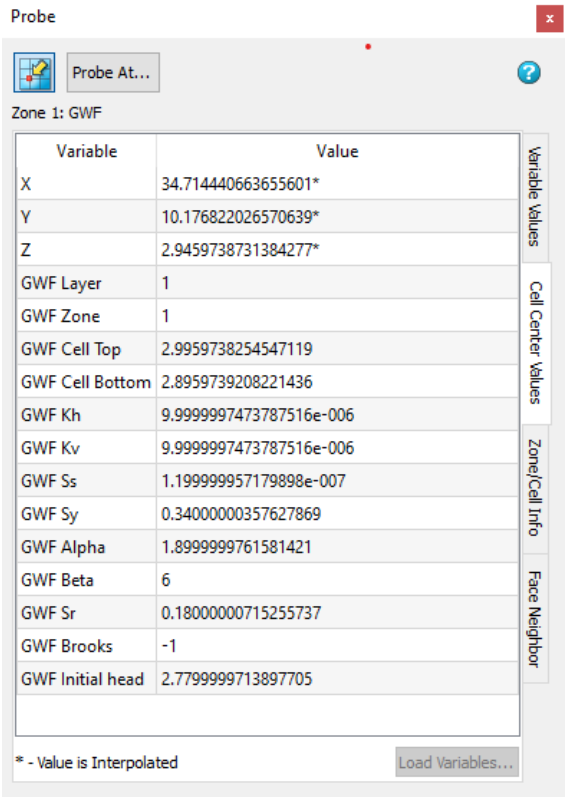


SWF results were returned because theSWF frame is at the front of the frame stack.

- In order to probe theGWF domain we have to move it to the front of the stack:



Here we see the results of probing a location in theGWF domain:


Probe
Probe At...
Zone 1: GWF
Variable
Value
X
34.7144406636556O1*
Y
10.176822026570639*
Z
2.9459738731384277*
GWF Layer
1
GWF Zone
1
GWF Cell Top
2.9959738254547119
GWF Cell Bottom
2.8959739208221436
GWF Kh
9.9999997473787516e-006
GWF Kv
9.9999997473787516e-006
GWF Ss
1.199999957179898e-007
GWF Sy
0.34000000357627869
GWF Alpha
1.8999999761581421
GWF Beta
6
GWF Sr
0.18000000715255737
GWF Brooks
-1
GWF Initial head
2.7799999713897705
* - Value is Interpolated
Load Variables...
Variable Values
Cell Center Values
Zone/Cell Info
Face Neighbor

Appendix C

Microsoft Access Useage

Index