Modflow User Tools (MUT) Version 1.22 User's Guide

August 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 It is not our intent here to support all existing Modflow packages, many of which have been superceded.*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, run MUT to produce a MODFLOW-USG^{Swf}-compatible data set and TECPLOT-compatible output files, and run TECPLOT to review the results of the model build.

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

Chapter 5 Examples Examples of MODFLOW-USG^{Swf} models built using MUT for both the verification and illustration of MUT and MODFLOW-USG^{Swf} useage.

Chapter 6 Tutorial In the tutorial, we

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 https://github.com/Grdbldr/MUT_Examples.git as shown in Figure 2.1. You do this by:

- Clicking on the green 'Code' button.

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

Note that the ZIP file also includes the executable for MODFLOW-USG^{Swf}.

Before you run MUT for the first time, you need to define a windows environment variable called USERBIN, as shown in Figure 2.2. You do this by:

- Typing the string 'en' in the windows taskbar search field.

- Opening the 'Edit the system environment variables' dialogue.

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

- Adding the variable USERBIN.

- Adding the path to USERBIN to the PATH variable.

Here we have set USERBIN to be equal to `c:\bin` but you are free to choose a different drive and folder. You will need to copy the files in _MUT_USERBIN folder (see panel 4 in Figure 2.1) to the USERBIN directory.

You should now be able to run MUT from the command prompt. To test this, start a new command prompt, then type `mut`, you should see the MUT header, shown in Figure 2.3. Type CTRL C to stop the program.

You can also run MODFLOW-USG^{Swf} by typing `usgs_1`. You should see the MODFLOW-USG^{Swf} header, shown in Figure 2.4. 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.
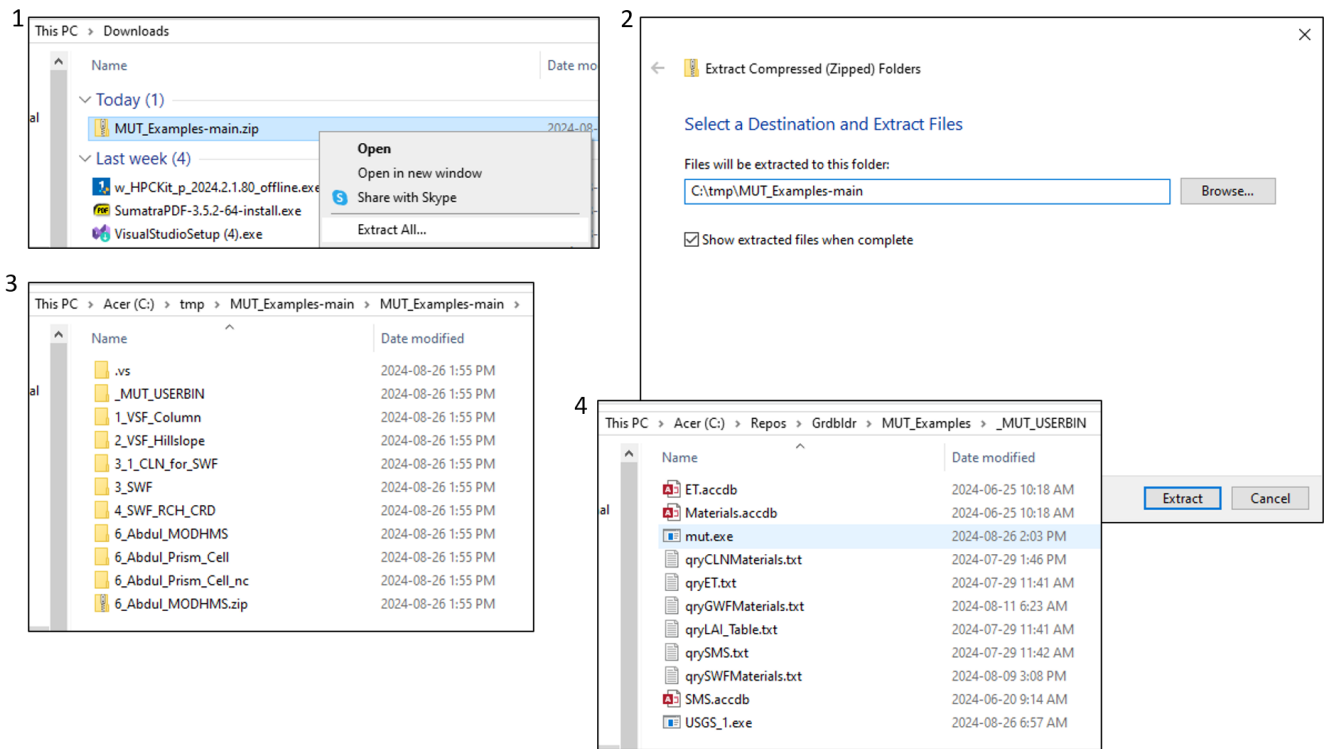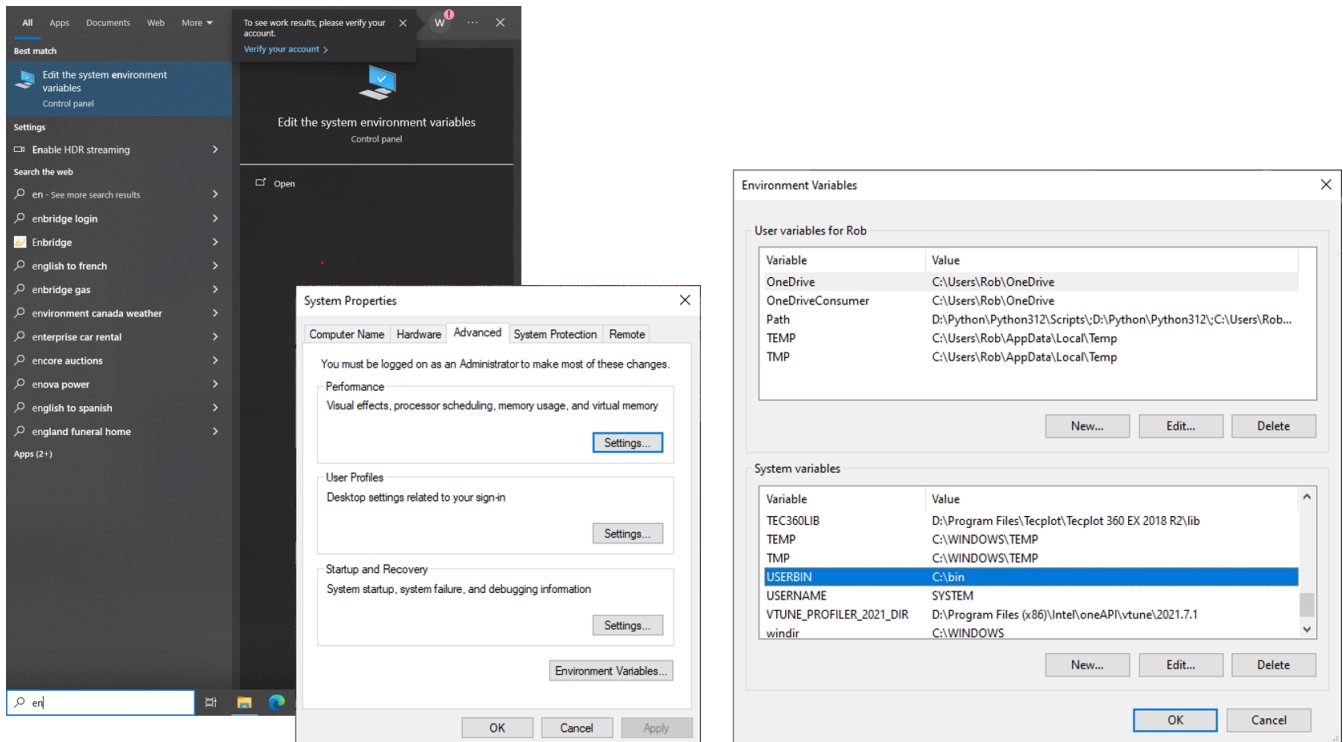
Figure 2.1: Downloading the MUT_Examples files

Figure 2.2: Defining a new environment variable

Figure 2.3: MUT Header

Figure 2.4: MODFLOW-USG^{Swf} header

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:
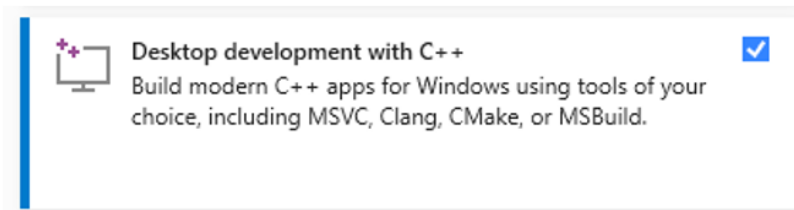
A free version of the latest INTEL FORTRAN compiler can be obtained from https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html. Once you are on the site just click the `Get It Now` button to download the Intel® HPC Toolkit, which includes INTEL FORTRAN. Choose the `Windows` 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 installion 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:
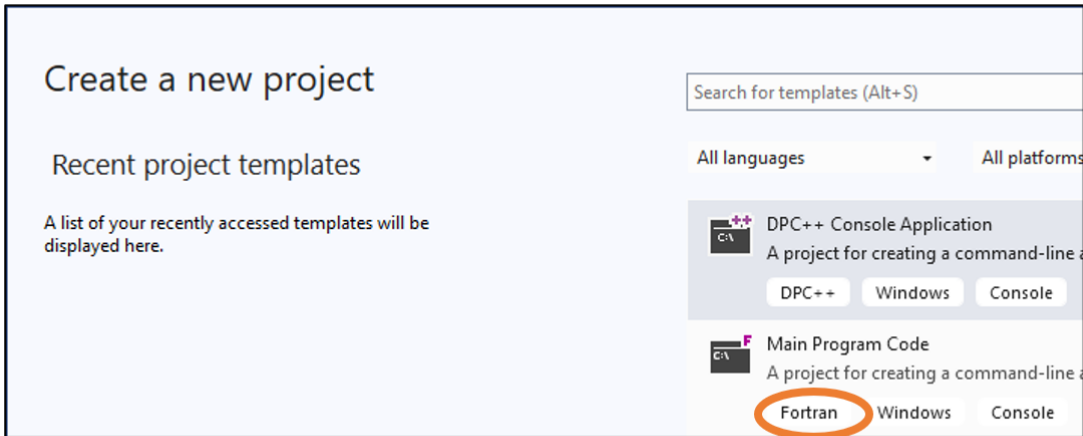
The MUT source files can be obtained from a GITHUB repository at https://github.com/Grdbldr/MUT_Source.git. Since GITHUB has been integrated into MICROSOFT VISUAL STUDIO we will use it to download the MUT repository. When you start MICROSOFT VISUAL STUDIO choose this option:
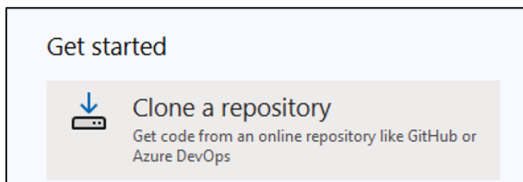
This opens the dialogue box shown below, where you can define the repository location on GITHUB and the path to the local repository. You can copy the link from the PDF file by right-clicking on it and choosing `Copy Link Adress`.
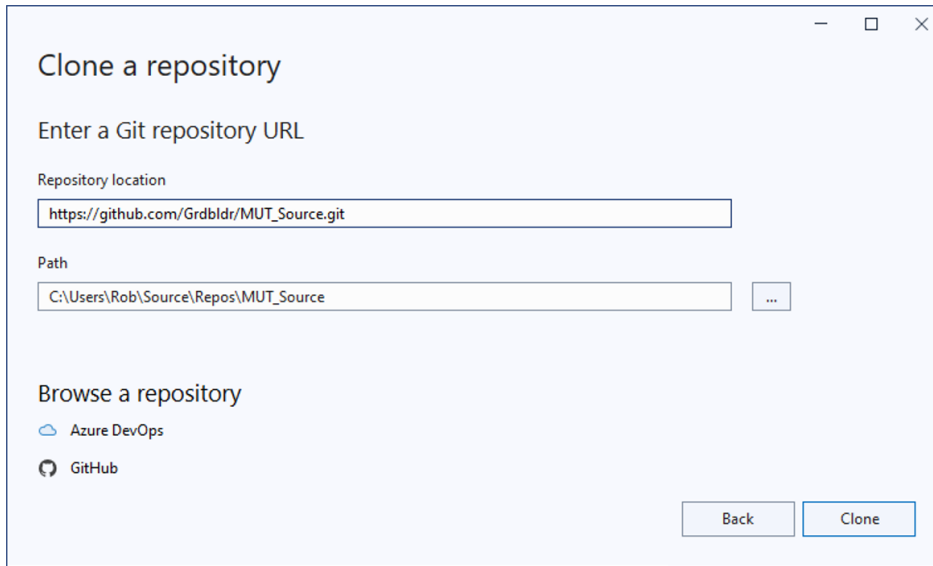
Now choose the `GitHub` option under `Browse a Repository` and you will see this dialogue shown below, Choose `Grdbldr/MUT_Source` from the list of repositories then click the `Clone` button.
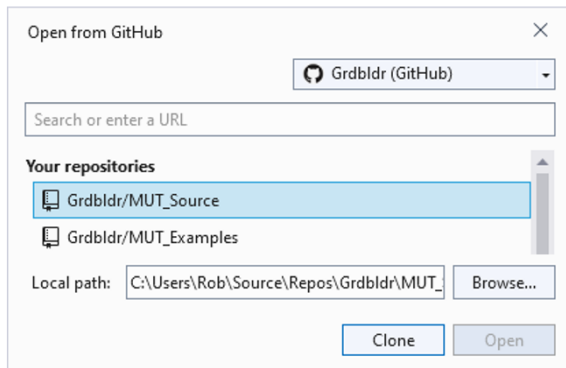
Figure 2.5 shows MICROSOFT VISUAL STUDIO after cloning `Grdbldr/MUT_Source`. Note the GITHUB dialogue on the right side of the changes tab, with information along the bottom about the repository. Details about using GITHUB in MICROSOFT VISUAL STUDIO are given in Tutorial 6.2.

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
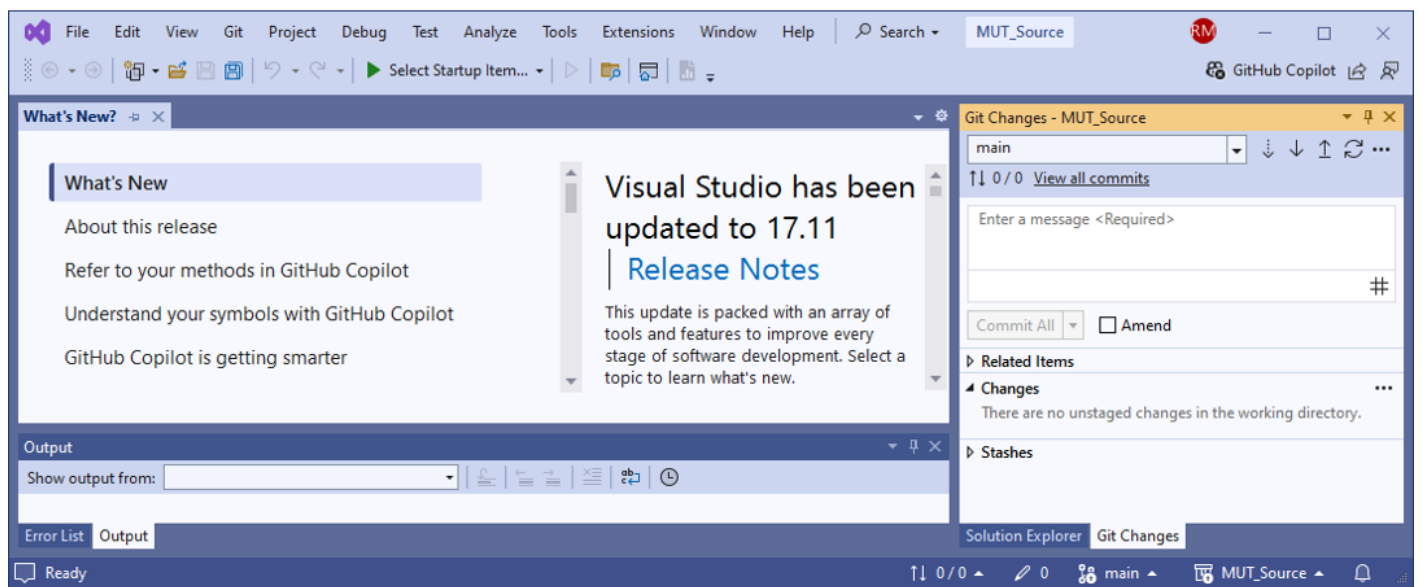
Figure 2.5: Microsoft Visual Studio after cloning Grdbldr/MUT_Source

Chapter 3

Model Build Workflow

This model build workflow is designed to create and fine-tune a MODFLOW-USG^{Swf} model, which represents a conceptual model we have of some hydrogeologic flow system, real or imaginary. MUT provides the link between conceptual model and MODFLOW-USG^{Swf} model, and is intended to minimize the amount of time we spend building and testing it.

The first step in any workflow is to define the conceptual model, which defines the model extents, inflows and outflows and material distributions and physical properties. To illustrate a model build, we will refer to the conceptual models developed for our existing suite of verification and illustration examples from Chapter 5.

Next we need to develop a MUT input file, which is a plain ascii text file that you can edit with your preferred editor (e.g. Windows Notepad). [1] Each modflow 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.

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:
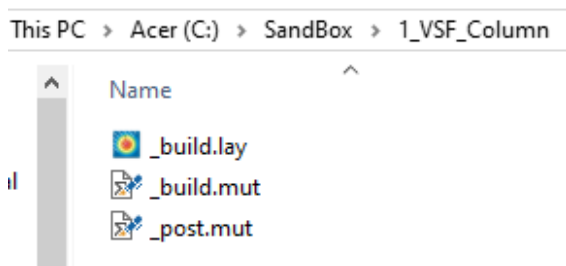
In this example, there are two MUT input files, one for the model build called _build.mut, one for post-

---

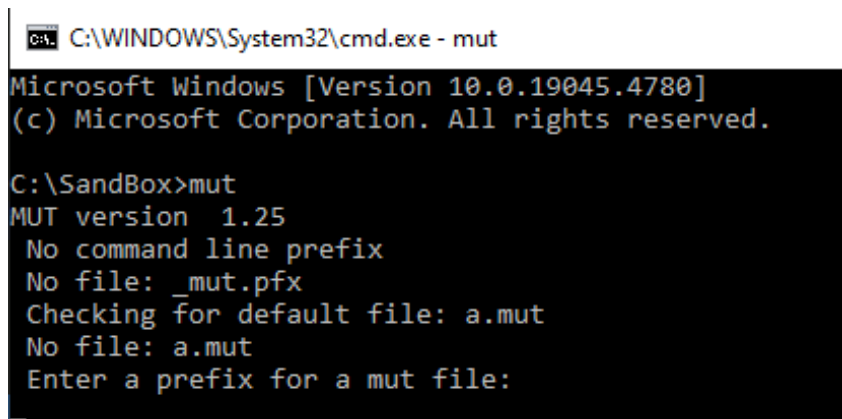[1] Our personal favourite editor is WinEdt (https://www.winedt.com/snap.html), which also provides a nice LaTeX document development environment when coupled with the TeX software package MiKTeX. This manual was produced using these word processing tools.

processing called _post.mut (discussed later in chapter 4) and a TECPLOT layout file called _build.lay used to visualize the model build results.

MUT tries to obtain a prefix in one of these four ways:

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:

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':

13

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


```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\SandBox\1_VSF_Column>_
```
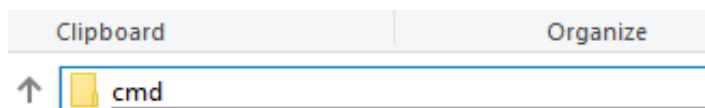
Now run MUT using the input file _build.mut by typing:

```
mut _build
```

As MUT processes the input file, note these features:

- Output is written to both the screen and to the file _buildo.eco as execution progresses. The first thing MUT writes is the version number, then the formal header, which also contains the build date.

- As comment lines are stripped from the input file, they are echoed to the screen and eco file and can provide a synopsis of the input file contents.

- Several new output files are created:

This PC › Acer (C:) › SandBox › 1_VSF_Column

_build.lay	Modflow.gwf.gsf
_build.mut	Modflow.lpf
_buildo.eco	Modflow.nam
_buildo.input	Modflow.oc
_buildo.Modflow.GWF.tecplot.dat	Modflow.rch
_buildo.Modflow.GWF_CELLS.tecplot.dat	Modflow.sms
_buildo.Modflow.GWF_CHD.tecplot.dat	Modflow.swf.gsf
_buildo.Modflow.GWF_RCH.tecplot.dat	scratcho.TECPLOT_GWF.bat
_buildo.TMPLT.tecplot.dat	scratcho.TECPLOT_GWF.mcr
_post.mut	scratcho.TECPLOT_GWF.neighbours.dat
batch.log	scratcho.TECPLOT_GWF.tecplot.dat
Modflow.bas	scratcho.TMPLT.bat
Modflow.chd	scratcho.TMPLT.mcr
Modflow.cln.gsf	scratcho.TMPLT.neighbours.dat
Modflow.dis	scratcho.TMPLT.tecplot.dat

- If we start the model build prefix with the underscore character, build output files, which have the prefix _buildo, appear near the start of the list if sorted by name. The build output contains several TECPLOT output files that 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 prevents confusion that can arise when out-of-date output files are 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 subtask has instructions that are used to define 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

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

---

[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.

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. **$_LengthUnit** The desired unit 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 **$_LengthUnit**.

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. no 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. **$_TimeUnit** 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 **$_TimeUnit**.

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:
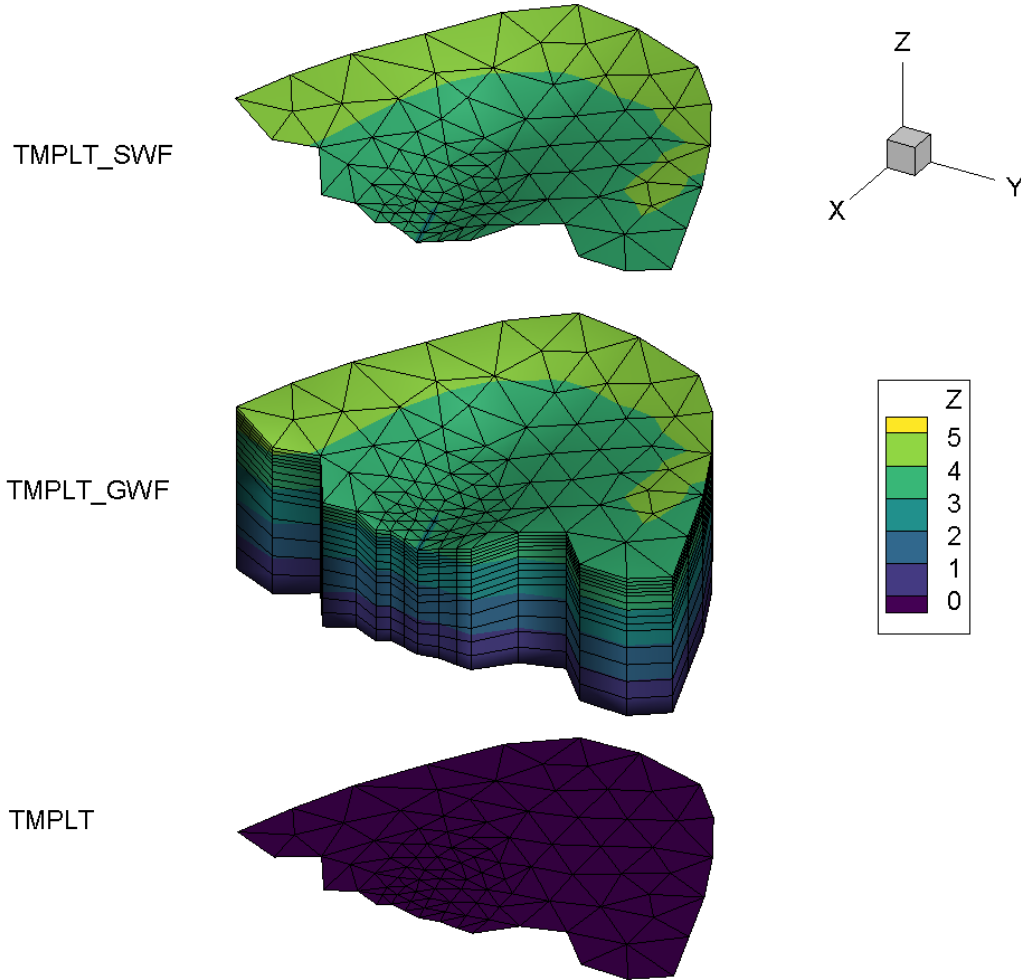
Some key features of this example are:

- 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.

---

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

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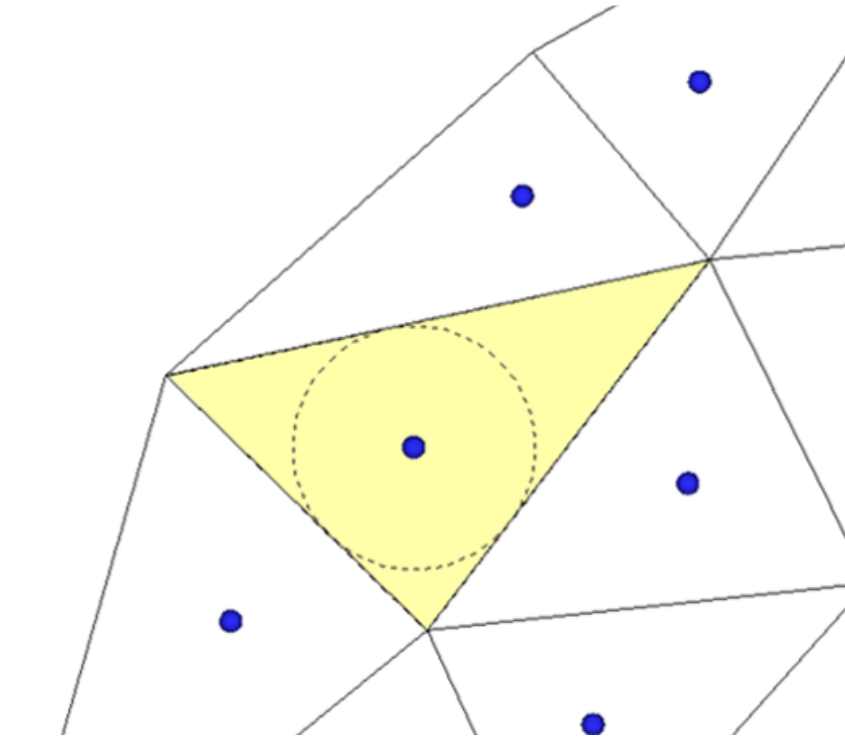
There are two ways that MODFLOW cell control volumes can be defined from the template mesh. By default, MUT uses a mesh-centred approach as shown here for a triangular-element template mesh:

---

[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:
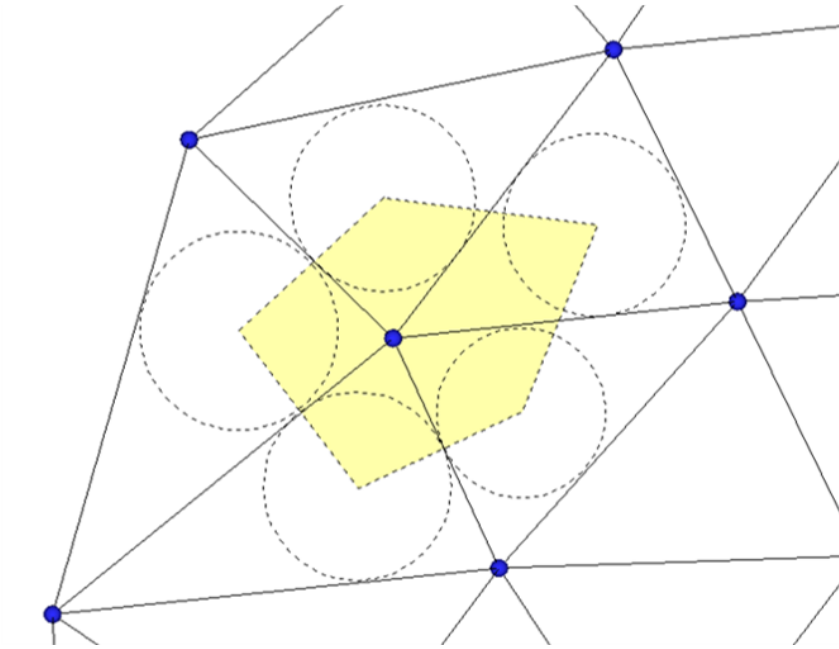
Some key features to note are:

- 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

Should add some into material about GWF build.

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.

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:
>
> - 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.

One 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 [6] 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.

- 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.

---

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

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, y coordinate pair.
>
> 2. ...
>
> 3. **r_x(n)**, **r_y(n)** nth x, y coordinate pair.
>
> This subtask reads a list of xz-coordinate pairs until an end instruction is encountered e.g.
>
> ## end elevation from xz pairs

A sample showing the use of this instruction [7] is shown here:

```
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.

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

---

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

- 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 that are given for the `top elevation` instruction.

Be 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.

Assigns the 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**.

24

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 `gs.asc`:

By default, element zone numbers reflect the horizontal layers added during mesh generation. If the template mesh is divided into horizontal patches with unique zone numbers, these can be assigned instead to the 3D GWF mesh [8] 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**

```
generate layered gwf domain

    top elevation
        elevation constant
        100.0
    end top elevation

    new layer
        layer name
        Top layer

        uniform sublayering
        100

        elevation constant
        0.0
    end new layer
```

---

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

```
end generate layered gwf domain
```

The generate layered gwf domain instruction is a subtask so it has a corresponding end generate layered gwf domain instruction. The GWF domain is generated from the top layer down, and the first instruction, top elevation, defines the ground surface elevation at the layer of elements in the TECPLOT_GWF mesh. It must begin with a top elevation instruction and at least one new layer instruction. Not shown in this example is an optional additional instruction, zone by template.

3.3.2   Constant Head(CHD)

```
Pre-requisites:
Activate one of \gwf,\swf\ or\cln\ domains
Choose cells

Instructions:
gwf constant head

Inputs:
Head    L
```

All chosen cells will be assigned a constant head equal to the specified total head value.

3.3.3   Drains(DRN)

```
Pre-requisites:
Activate one of \gwf,\swf\ or\cln\ domains
Choose cells

Instructions:
gwf drain

Inputs:
Drain conductance  L/T
```

All chosen cells will be assigned a drain elevation equal to the top elevation of the cell with the specified drain conductance.

3.3.4   Recharge(RCH)

```
Pre-requisites:
```

```
Activate one of \gwf,\swf\ or\cln\ domains
Choose cells

Instructions:
    gwf recharge
    0.4      ! recharge
    3        ! recharge option

Inputs:
Recharge rate   L/T
Recharge Option  -
```

All chosen cells will be assigned a recharge.

3.3.5 Pumping

3.4 Connected Linear Networks(**CLN**) Domain

3.5 Surface Water Flow(SWF) Domain

In MUT a 2D template mesh is first defined and then a surface water flow (SWF) domain is generated from it using the instruction generate swf domain, as shown in this example:

```
generate swf domain
    top elevation
        elevation from gb file
        ./gb/grid.nprop.Surface elevation
    end
end
```

3.5.1 Critical Depth

3.5.2 Zero-Gradient Depth

Chapter 4

Model Execution and Post-Processing

Chapter 5

Examples

5.1 Verification

5.1.1 Unsaturated flow in a 1D 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 HydroGeo-Sphere at 20 and 40 seconds:

Flow in a 1D column
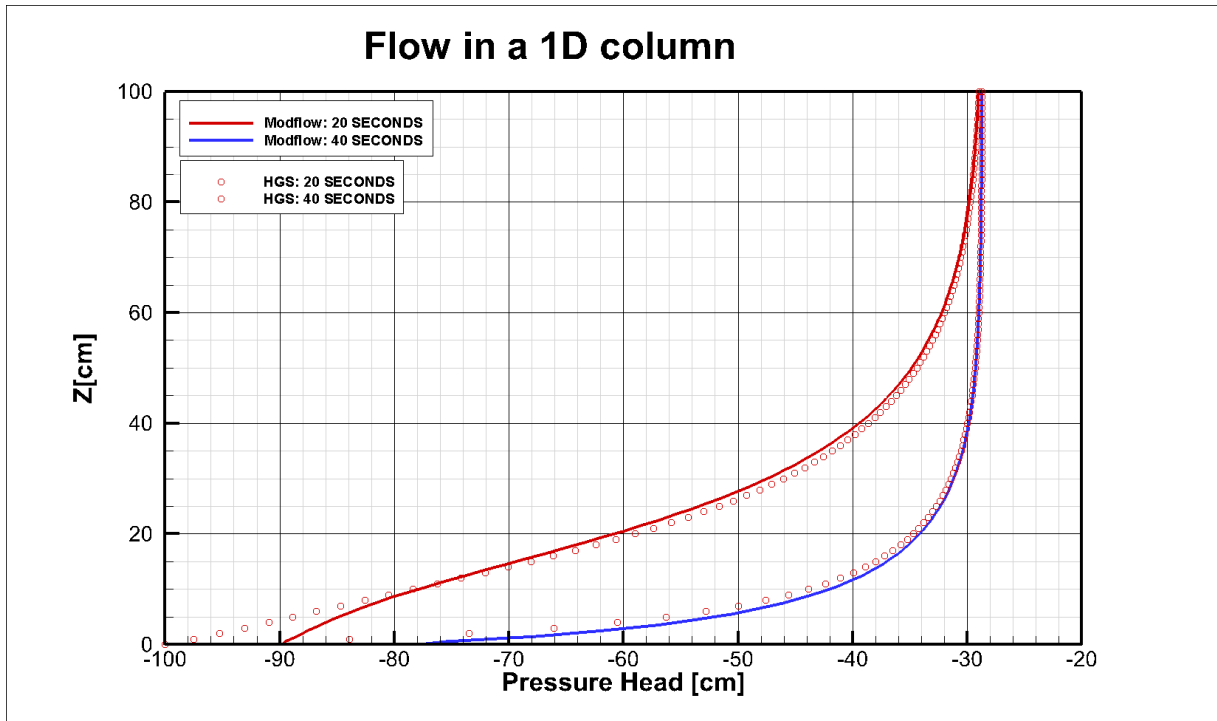
Here is a comparison of saturation versus depth results for Modflow-Usg^{Swf} and HydroGeoSphere at 20 and 40 seconds:

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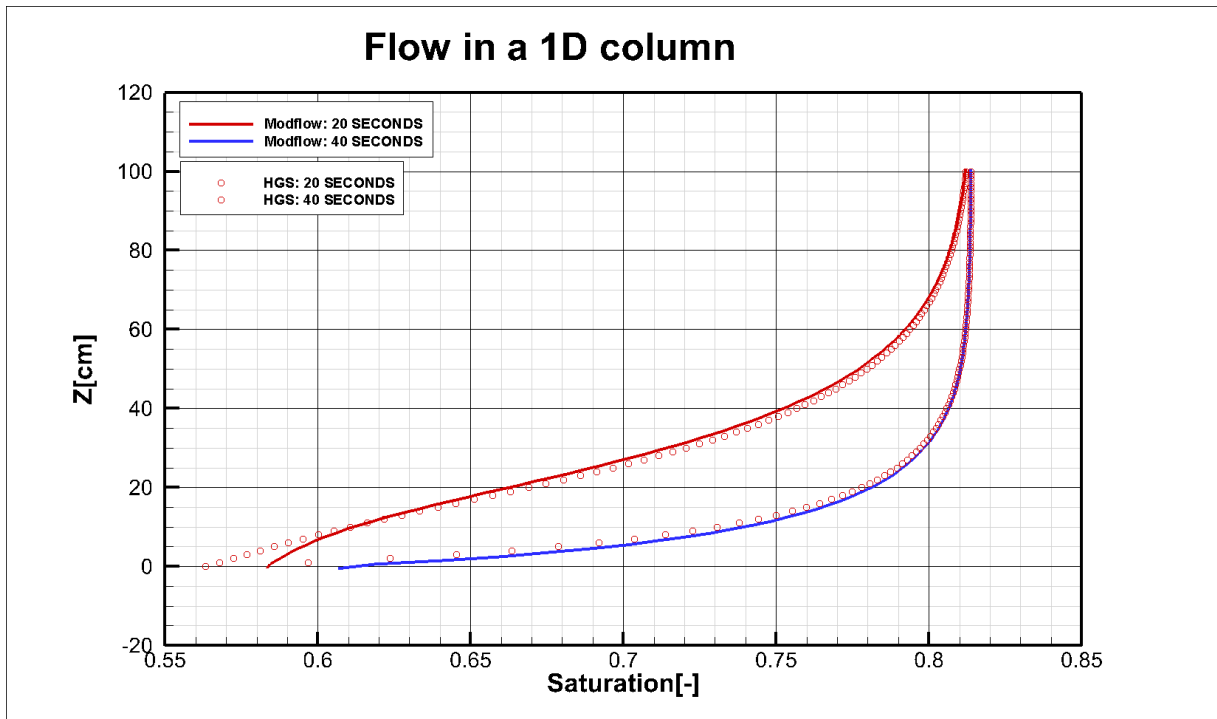
Although these results look good, there is a discrepancy between the unit systems used in the two models (e.g. K is 10 in HYDROGEOSPHERE and 1e-5 in MODFLOW-USG^{Swf}. HGS column height appears to be 100 cm, while MODFLOW-USG^{Swf} model is 100 m thick. MUT needs to be modified to allow the user to change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

5.1.2 Unsaturated flow in a 2D 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`
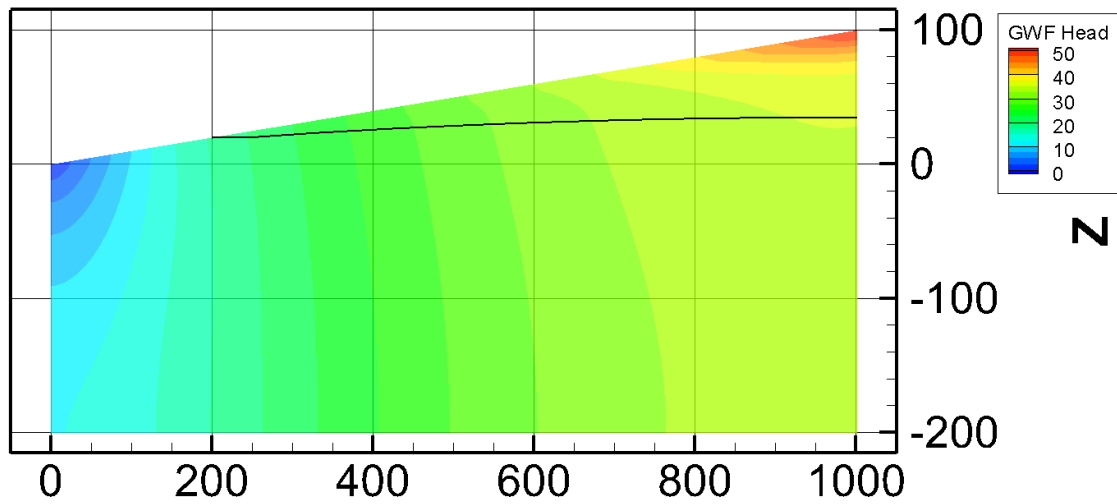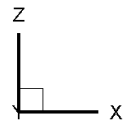
Some key features of this example are:

- 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.

5.1.3 Abdul's Experiment: 3D Unsaturated GroundwaterFlow

Properties from the HGS manual are shown in Table 5.2.

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

5.2 Illustration

33

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

Tutorial

6.1   Mut Useage

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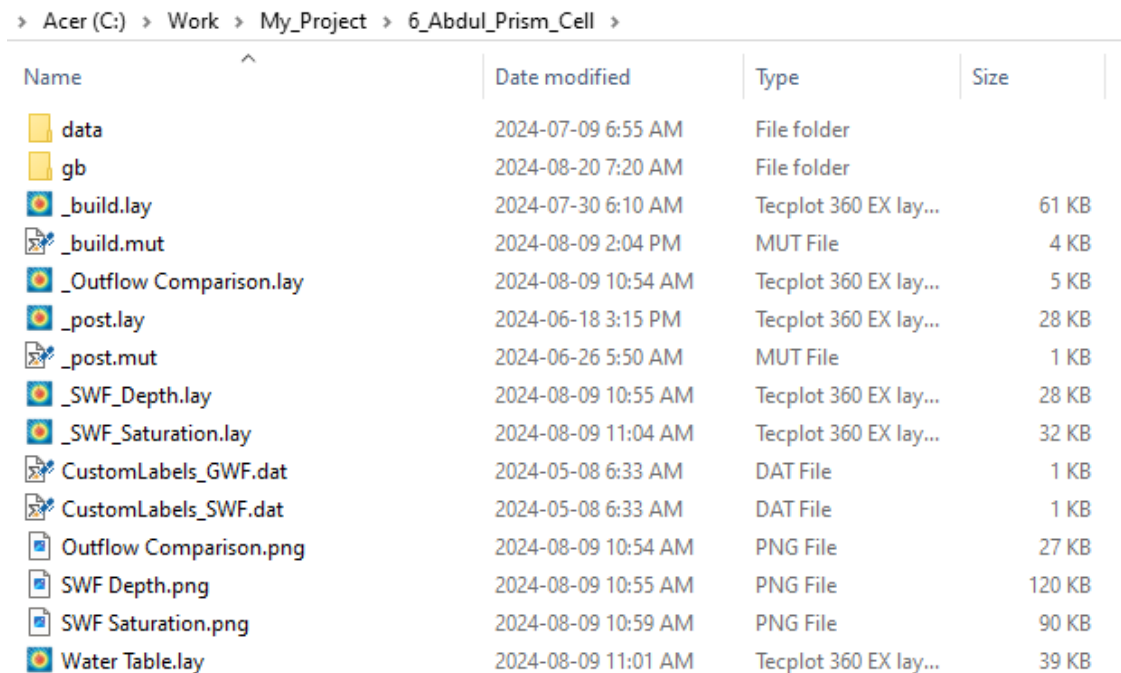
Output control Instructions defining a list of output times at which Modflow output files (e.g. heads, drawdowns, cell-by-cell flows etc.) are to be written.

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 6.1 shows the contents of our `6_Abdul_Prism_Cell` folder. Yours may look different depending on the root drive and folder location.

Figure 6.1: 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:
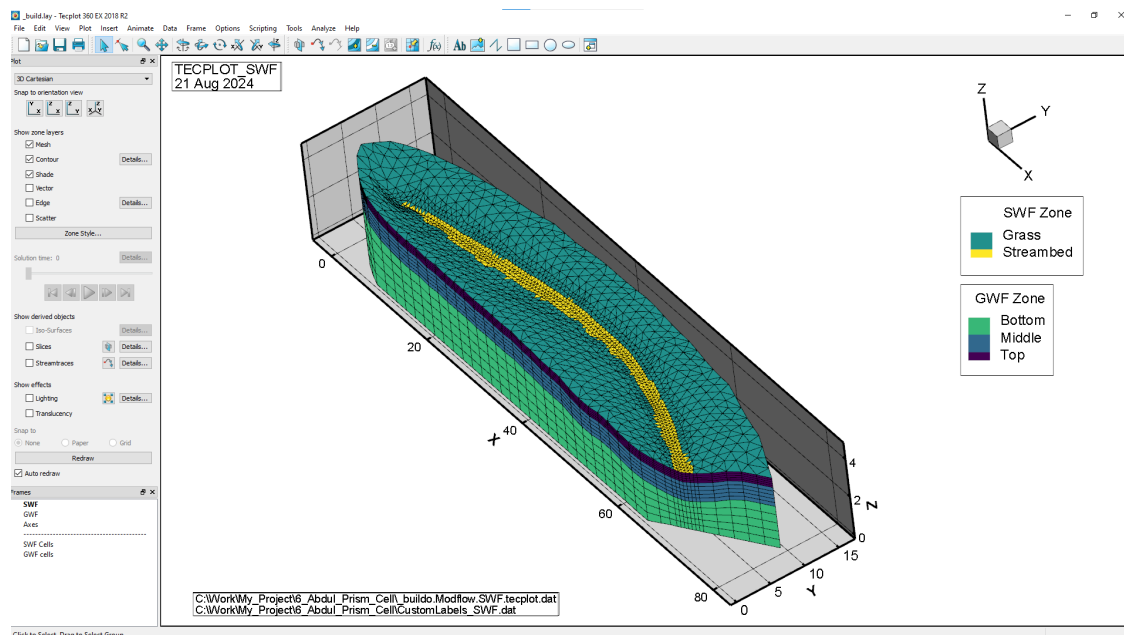
```
mut _build
```

Step 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:
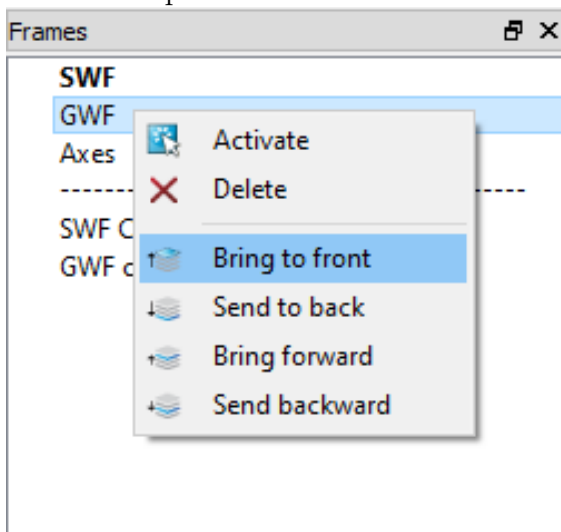
- A TECPLOT window should open:

  This TECPLOT layout file has been constructed with multiple frames (see lower left 'Frames' window) showing details about the SWF and GWF model domains. This default view shows the distribution of the various materials defined in the model, such as the SWF 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 the SWF domain:
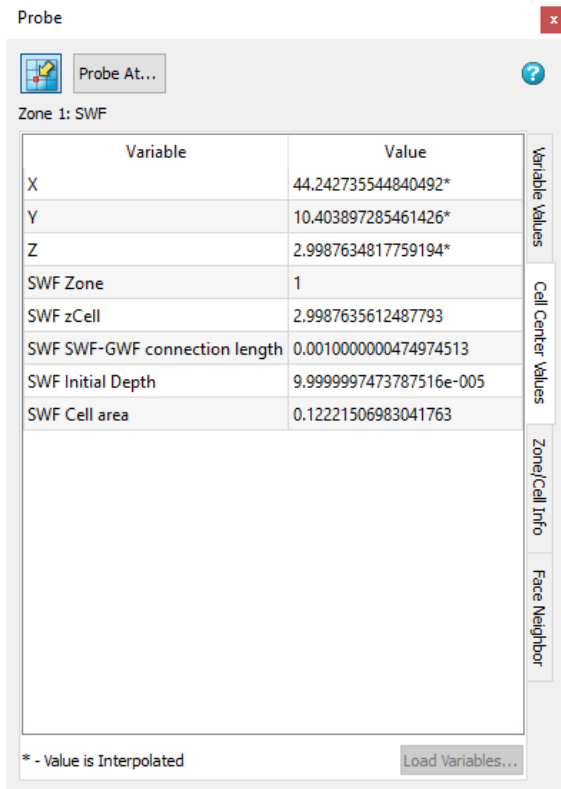
SWF results were returned because theSWF frame is at the front of the frame stack.

- 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:

Step 5: Run Modflow to Generate Output

Step 6: Run Mut to Post-Process the Modflow Output

Step 7: Run Tecplot to Visualize the Modflow Output

6.1.1 Suggested Workflow

A well-designed workflow should minimize the introduction of human error into the modelling process and facilitate later review by senior modellers. Below we describe one possible approach that can be used as a starting point for implementing your own personal workflow. We will use the verification example `6_Abdul_Prism_Cell` to demonstrate our suggested workflow. The steps in the workflow are:

1. Copy an existing MUT project folder to a new working folder.

2. Modify the `_build.mut` 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. Repeat steps 2-3 until the new project is defined correctly.

5. Run Modflow to create the new project output files (e.g. time-varying hydraulic head, drawdown etc).

6. 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.

7. Run TECPLOT and examine the Modflow output files.

6.1.2 Mut Input File Structure

MUT recognizes files which have the extension `.mut` as input files and reads and interprets them to produce both MODFLOW-USG^{Swf} output files and TECPLOT input files.

Comments begin with an exclamation point character ! and are ignored by MUT. MUT initially strips the input file of all comments and creates a clean copy called *prefix*`o.input`, which is then processed by MUT. This means comments can be placed anywhere in the input file.

6.2 GitHub Useage in Microsoft Visual Studio

6.3 Tecplot Useage

Index