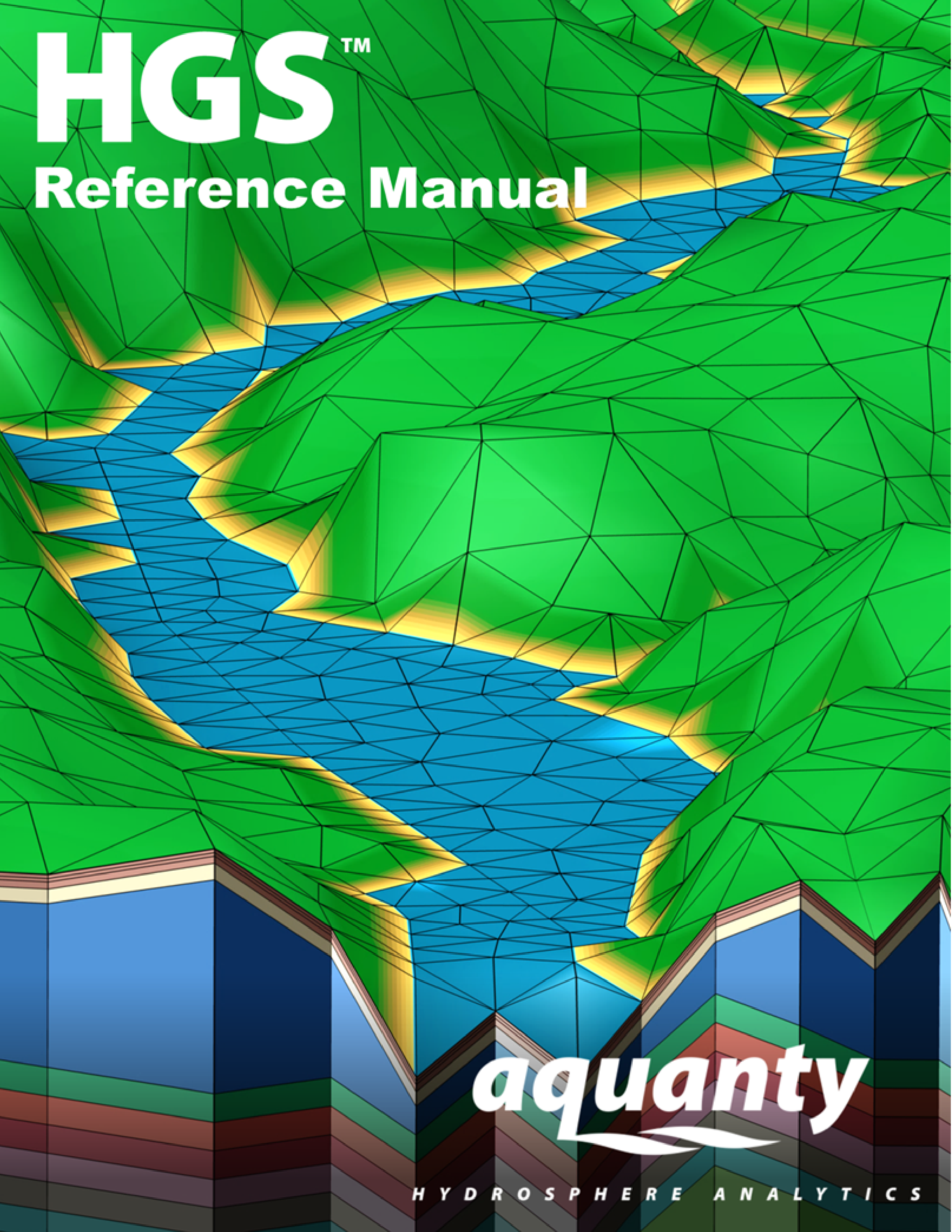
# HGS™ Reference Manual



aquanty

HYDROSPHERE ANALYTICS

HGS Simulations is a product of Aquanty Inc.
600 Weber Street North, Unit B
Waterloo, Ontario N2V 1K4

This manual documents HydroGeoSphere (HGS) version 2778. Released: February 3, 2025.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Quick Start Guide

The goal of this guide is to get you up to speed quickly on the basic operation of **HydroGeoSphere** (HGS). The topics covered include:

- HGS installation.
- A brief overview of the key executable files.
- Running a model.

## 1.1 HGS Installation

**HydroGeoSphere** consists of a suite of 64-bit applications that target either the Windows (Windows 10 and newer) or Linux operating systems. In order to run HGS, a valid HGS license file `hgs.lic` must be present in the installation directory. To obtain a valid HGS license file, please email the `hostid.txt` file located in the installation directory to `info@aquanty.com`.

### 1.1.1 Windows

By default, HGS is installed in the directory:

`C:\Program_Files\HydroGeoSphere`

The installation folder contains the key executable files (`grok.exe`, `phgs.exe`, `hsplot.exe`, `hgs2vtu.exe`) required for simulation and results post-processing, the `hostid.txt` file, and a number of DLL files. Folders within the installation directory include:

- *docs*: This folder contains the theory, reference, and verification manuals which are important references for model setup.
- *illustration* and *verification*: These folders contain example problems, many of which are discussed in the verification manual.

Note that the installer will attempt to add the installation directory to your system path. Should it fail to do so, for example, if it does not have write access, then you will need to update your system path manually. We recommend that you always run the installer with administrator privileges if possible.

### 1.1.2 Linux

In Linux, HGS is installed under the current folder in the directory:

`HydroGeoSphere-RevNum-Linux`

where "RevNum" is the revision number of the installation. The installation folder contains the key binary files (`grok`, `hgs`, `hsplot`, `hgs2vtu`), the `hostid.txt` file, and the EULA. Folders within the installation directory include:

- *docs*: This folder contains the theory, reference, and verification manuals which are important references for model setup.
- *lib*: This folder contains the external dependency library files.
- *verification*: This folder contains example problems, many of which are discussed in the verification manual.

The installer modifies the `~/.bashrc` file by adding the new environment variable `HGSDIR` that points to the HGS installation directory, the HGS installation directory to `PATH`, and the `lib` directory to `LD_LIBRARY_PATH`. Users are required to either start a new shell or run `source ~/.bashrc` for these changes to be added to their environment.

Some notes to keep in mind:

- Users are encouraged to check the contents of their `~/.bashrc` file to ensure their bash environment is set up appropriately for their own computing purposes and for running **HydroGeoSphere**.
- Users on other Linux terminal shells (e.g., zsh, csh, ksh, etc.) should maintain their own shell profiles to appropriately set their `HGSDIR`, `PATH`, and `LD_LIBRARY_PATH` environment variables, if necessary. Please consult with your local system administrator if you are unsure about how to do this for your HPC computing environment.

## 1.2 Key Executable Overview

There are three steps required to set up, simulate, and view the results of a simulation.

1. A data file is prepared for the pre-processor (called **grok**) which is then run to generate the input data files for HGS.

2. HGS is run to solve the problem and generate output data files.
3. Depending on the problem, post-processing of the data is completed using **HSPLOT** or **HGS2VTU**, to convert the data into a format for visualization and analysis.

### 1.2.1 Grok (`grok.exe`)

The **grok** input file contains all of the information and instructions required for the HGS simulation. The **grok** file name consists of a meaningful prefix (up to 40 characters) to which the extension `.grok` is appended. For example, if the problem prefix created by the user is *test*, then the general input file created by the user will be `test.grok`. **Grok** will attempt to read the problem prefix from the `batch.pfx` file, which contains a single line with the prefix name. If **grok** is unable to find this file, then the user will be prompted to enter the prefix name at the console. Information contained within the **grok** file includes the mesh definition, model parameterization, initial conditions, boundary conditions, convergence criteria, and simulation output criteria. The pre-processor, **grok**, performs its task in the following order:

1. Read and allocate default array sizes.
2. Read the problem identification information.
3. Read instructions for generating the grid.
4. Perform grid modifications if necessary.
5. Generate default properties for all parameters.
6. Read optional instructions for modifying the default parameters.
7. Write the HGS-compatible data types.

Once the *prefix*`.grok` file has been built by the user it is compiled by running `grok.exe`. A more detailed description of **grok** and its associated commands are contained in Chapter 2. We note that `.grok` files in the *illustration* and *verification* folders are an excellent resource for reviewing **grok** structure and the use of **grok** commands.

### 1.2.2 HGS (`phgs.exe`)

After the execution of `grok.exe`, which writes all the HGS-compatible data files, `phgs.exe` is executed to perform the model simulation. There is little user involvement at this stage other than the configuration of the parallel execution details as discussed in Section 2.5.9.

### 1.2.3 HSPLOT (`hsplot.exe`)

The executable `hsplot.exe` is used the post-process the simulation results for viewing in Tecplot (Appendix D). **HSPLOT** can be executed during an HGS run or following its completion. The resulting output files (*prefix*`o.`*domain*`.dat`) can be opened in Tecplot to view the simulation results in three dimensions.

### 1.2.4 HGS2VTU (`hgs2vtu.exe`)

The executable `hgs2vtu.exe` is an alternative to `hsplot.exe` that can post-process the simulation results into various output formats (Appendix E). By default, unstructured UGRID VTU output files are generated, which can be opened with ParaView to view the simulation results in three dimensions.

## 1.3 Running a Model

We conclude this chapter by describing the steps to run the Abdul model problem, the model files for which can be found in

`C:\Program_Files\HydroGeoSphere\verification\abdul`

For additional details on this problem refer to Section 1.4.1 of the Verification Manual `hydrosphere_verif.pdf`. The steps to run this model problem are as follows.

1. Open a command console in the folder:

   `C:\Program_Files\HydroGeoSphere\verification\abdul`.

2. Run `grok.exe`.

3. Run `phgs.exe`.

4. Run `hsplot.exe` or `hgs2vtu.exe`.

5. Open the output files to view the simulation results.

You can check for successful completion of **grok**, **HydroGeoSphere**, **HSPLOT** or **HGS2VTU** from the command line by checking the returned exit code. An exit code of 0 means the executable completed without any errors, whereas a nonzero exit code indicates that an error occurred. In most cases, an error will be accompanied by an informative error message written to the console and the corresponding log file for the given executable.

Note that Windows users who receive a DLL error when running one of the executable files should copy the DLL files from the installation folder to the current simulation folder. Alternatively, Windows users can add the directory `C:\Program_Files\HydroGeoSphere` to their system path. Updating the system path makes it possible to run a model from any folder without copying any HGS executable files or DLL files to that folder and is the preferred method of operation.

# Chapter 2

# Input/Output Instructions

## 2.1 General

Before presenting in detail the input data needed for the numerical simulations, some general information about the format and nature of the input data is first given.

There are two steps involved in solving a given problem. First, a data file is prepared for the pre-processor (called **grok**[1]) which is then run to generate the input data files for **HydroGeoSphere**. Second, **HydroGeoSphere** is run to solve the problem and generate output data files.

The **grok** input file name consists of a meaningful prefix of up to 40 characters to which the extension `.grok` is appended. This prefix will determine the input and output filenames. The **grok** listing file name will be the problem prefix to which the letter `o` and the file extension `.eco` are appended. For example, if the problem prefix specified by the user is `test`, the general input file to be created by the user will be `test.grok` and the output listing, or echo, file generated by the pre-processor will be `testo.eco`. Some simulations will require more than one input file (e.g., initial heads read from file) and will result in the generation of more than one output file. As a rule, all input files needed during a specific simulation will have the problem prefix plus a given extension as filename while all generated output files will have the problem prefix, the letter `o`, plus a given extension as filename.

Throughout the manual, we will adopt the convention of using *italics* to indicate problem-dependent, user-defined portions of filenames (e.g., prefix, species name, etc.) and `typewriter font` to indicate invariant portions generated by **HydroGeoSphere**. For example, in the filename *prefix*`o.conc.`*species*`.0001` the *prefix* and *species* portions would be the user-defined

[1]grok /grok/, var. /grohk/ vt. [from the novel "Stranger in a Strange Land", by Robert A. Heinlein, where it is a Martian word meaning literally 'to drink' and metaphorically 'to be one with'] The emphatic form is 'grok in fullness'. 1. To understand, usually in a global sense. Connotes intimate and exhaustive knowledge. Contrast zen, which is similar supernal understanding experienced as a single brief flash. See also glark. 2. Used of programs, may connote merely sufficient understanding. "Almost all C compilers grok the void type these days."

prefix and name of a solute, or species, while the `o.conc.` and `.0001` portions would be generated by **HydroGeoSphere** automatically.

After the pre-processor starts executing, it prompts the user to enter the prefix for the problem interactively from the keyboard. For cases in which the same input file is being used repeatedly, you can create a file called `batch.pfx` which consists of a single line that contains the problem prefix. If the file is present, the prefix will automatically be read from the file and you will not be prompted to enter it from the keyboard. This file should be placed in the same directory as the *prefix*`.grok` file.

Briefly, the pre-processor performs its tasks in the following order:

1. Read and allocate default array sizes.
2. Read problem identification information.
3. Read instructions for generating grid.
4. Perform grid modifications if necessary.
5. Generate default properties for all parameters.
6. Read optional instructions for modifying the default parameters.
7. Write the **HydroGeoSphere**-compatible data files.

Tasks 3 and 6 are guided by instructions issued by the user in the *prefix*`.grok` file. The generation of a complete set of default data by Task 5 tends to minimize the amount of data which must be supplied by the user.

Here is an example instruction and some input data that illustrates some common conventions that will be used throughout the manual:

## Example instruction text

1. **xlen, nbx** Domain length [L] and number of blocks in the $x$-direction.
2. **xi(i), i=1,nx** Nodal $x$-coordinates [L].
3. **inode(i)...end** Node numbers.

●●●

The pre-processor instruction is separated from the preceding text by a horizontal line, and is written using the sans serif font. It must be typed in the *prefix*`.grok` file *exactly as shown*, with the exception that it is not case-sensitive, and blanks before and after the instruction are optional. Note that only one blank is allowed between any two words in an instruction.

If the instruction requires input data, there will follow a series of numbered lines, each containing boldfaced **variable names** and a description of what is to be read. Each numbered line will correspond to one *or more* input parameters.

Usually, the number of items required in the data file are indicated by how many boldfaced variable names are present on the line. The data type of an input parameter should be clear from its context, e.g., the size of an array is an integer, whereas hydraulic conductivities are real valued. Numerical values are read in free-format so integers and reals do not need to be lined up in columns and they can be separated by blanks or commas. A descriptive comment can be included inline after the last data value has been read from the line, but should be avoided when reading character strings (e.g., filenames). Character strings, such as filenames, may be arbitrarily long unless a maximum length is specified in the command description.

In this example, three items of input are required. The first item **xlen, nbx** requires that the user enter a real value (i.e., domain length) followed by an integer value (i.e., number of blocks) on the first non-blank or uncommented line following the instruction. Note that the comma between the two input parameters is entirely optional and is not required when parsing the `.grok` file.

The second item **xi(i), i=1,nx** reads **nx** values into the array **xi**. The size of **nx** is problem dependent (e.g., number of nodes in $x$, number of species, etc.) and it is up to the user to supply enough values to satisfy the read statement. Array input typically expects one value per line, however, there are exceptions to this rule. In cases where inline input is available or required, it will be made clear in the command description with an illustrative example. If values are entered on one line, they must be separated by either spaces or commas. A common use of array based input is to specify a table of values, for example, **time(i), value(i), i=1,n**, which would be formatted as follows

```
time(1) value(1)
time(2) value(2)
  :        :
time(n) value(n)
```

Naturally, this input format can be extended to tables with more than two columns.

Finally, the third item **inode(i)...end** indicates a list, in this case of node numbers, that is to be read until an end instruction is encountered. The list values must be entered one per line. A common use of list based input is to specify a table of values, for example, **time(i), value(i)...end**, which would be formatted as follows

```
time(1) value(1)
time(2) value(2)
  :        :
time(n) value(n)
end
```

Naturally, this input format can be extended to tables with more than two columns.

The end of the documentation that pertains to a specific instruction is designated by three dots: ● ● ●.

So for this example instruction, assuming that **nx** is equal to 5, the following statements in the *prefix*`.grok` file would satisfy the input requirements:

```
Example instruction text
10.0    100
0.0   2.0   4.0   6.0   8.0   10.0
1
2
3
5
6
end
```

In some cases (there are not too many) an instruction will have a more complex input structure of the form **val(i,j), i=1,m, j=1,n**. The indices are always listed from fastest to slowest varying from left to right. Hence, this input would be written in a file as:

```
val11 val21 ... valm1
val12 val22 ... valm2
  :     :         :
val1n val2n ... valmn
```

If index $j$ was listed first followed by index $i$, then the input in the file would be transposed. In all cases, the instruction will contain a helpful example to show how the input should be formatted in the file.

Less commonly, input for an instruction will allow for one or more optional parameters, which are surrounded by braces '()' in the input format. For example, in the input **xlen, nbx, (x0)** the last value **x0** is optional. If not provided, it will obtain a default value specified by the instruction description. Optional parameters always appear at the end of an input sequence.

Some instructions are controlled by input routines that have their own subset of input instructions, some or all of which may be optional. For example, the instruction Solute...End is used to define a new solute and in its simplest form appears as:

```
Solute
end
```

In this case, the End instruction immediately follows the Solute...End instruction, and no optional instructions have been issued. The End statement is required so that **grok** knows when to exit the solute definition routine. Such instructions will be indicated using the following convention:

## Example instruction text...End

•••

where the text ...End indicates that the instruction (e.g., Solute...End) will be followed by optional instructions or input and terminated by an End instruction.

Before **grok** processes instructions contained in a *prefix*.grok or a material properties file (see Section 2.8.1.4) it first makes a working copy of the file in which any line that is completely blank or that begins with an exclamation point (!) is removed and in which the contents of any included file are copied. This allows you to include blank lines and comments when and where required to improve the readability and clarity of the input.

Included files can be used to avoid having to cut and paste or comment and uncomment large sections of input instructions. Long lists (e.g., of node numbers or boundary condition data) and cases where various different grid generation approaches are being tried are good candidates for application of the include feature. For example, if we wanted to use include to supply data to the example given above, we could use the following instruction in *prefix*.grok:

```
Example instruction text
10.0    100
0.0   2.0   4.0   6.0   8.0   10.0
include my.node_list
```

and where the file my.node_list could contain, for example:

```
1
2
3
5
6
end
```

If you now wanted to substitute another node list you could, for example, supply different node numbers in the file my_other.node_list and then just change the file name given in the include instruction. Included files can contain groups of instructions and input, or just bits of input for a single instruction. Include files can contain additional include instructions, which can be nested to an arbitrary depth.

As **grok** reads and processes the copy of the *prefix*.grok file it also creates the *prefix*o.eco file. Results of the **HydroGeoSphere** data generation procedures are written to this file so if there are any problems reported by the pre-processor you should check this file first to determine their nature and how you might fix them. If an error occurs while reading the input data, then **grok** will halt execution and issue an error message (to the screen and the *prefix*`o.eco` file) of the form:

```
GROK: 500

**************************************
*** INPUT ERROR, HALTING EXECUTION ***
**************************************

GENERAL PRE-PROCESSOR: Unrecognized instruction

1
```

In this case, the last instruction (i.e., `500`) has, for some reason, caused an error. You should now check the input files to further investigate the cause of the problem, starting with the *prefix*`.grok` and material properties files.

### 2.1.1 File Process Control Options

The following instructions control how the pre-processor treats instructions in the *prefix*`.grok` file and can be inserted at any point in the file and as often as required, except of course when input for a specific instruction is expected.

---

#### Skip on

With skip mode turned on, **grok** will read but not act on any subsequent instructions.

●●●

---

#### Skip off

Turns skip mode off, so **grok** will resume acting on instructions.

●●●

---

#### Skip rest

**grok** exits the loop for reading instructions from the *prefix*`.grok` file and proceeds to generate the **HydroGeoSphere** data files.

●●●

---

#### Pause

This instruction causes **grok** to pause at the current location in the *prefix*`.grok` file until the user presses a key.

●●●

### 2.1.2 User Defined Variables

This section describes pre-processor commands that can be used to define/undefine variables in your **grok** file and material properties files (see Section 2.8), similar to how variables are used in a batch script or shell script, albeit, on a much simpler level. The syntax of these commands is different from other **grok** commands you will encounter in this manual for two reasons:

1. To mimic the syntax for defining variables used by batch or shell scripts.
2. Because these commands are parsed by **grok** during scratch file generation and are not actually present in the final **grok** or material properties files.

We begin by describing how to define a new variable or overwrite the value of an existing one:

```
set variable $<varname>=<value>
```

The variable name (`$<varname>`) may consist of up to 256 characters, is case insensitive, and must adhere to the following rules:

1. Contain at least two characters.
2. The first character must be the dollar sign ($), which is a special character reserved for identifying pre-processor variables.
3. The second character is a letter or underscore.
4. All remaining characters are letters, numbers, or an underscore.

The variable's value (`<value>`), which is always treated as a string, may consist of up to 4096 characters and must not contain a dollar sign ($) character. The `set variable` command ignores any leading/trailing whitespace around the variable name and its value. Inline comments are also ignored. For example, the following commands are equivalent:

```
set variable $path=C:\my_file_path    ! inline comment
set variable   $path =C:\my_file_path
set variable $path=   C:\my_file_path
set variable   $path  =  C:\my_file_path
```

Each command defines the variable `$path` to have the value `C:\my_file_path`. If you wish to retain leading whitespace in the variable value, then you can do so by enclosing it in double quotes (“”). For example, the command

```
set variable $path="   C:\my_file_path"
```

assigns to the variable `$path` the value `   C:\my_file_path`. Note that the enclosing double quotes are automatically stripped from the variable value by the `set variable` command. If the value is a file path with spaces, for example, and you would like to retain the enclosing double quotes then use

```
set variable $path=""C:\my file path with spaces""
```

If you would like to assign an empty value to a variable, then you can do so as follows using either of the equivalent commands:

```
set variable $<varname>=
set variable $<varname>=""
```

If you use the `set variable` command without any parameters, then a list of all currently defined variables and their values will be written to the console. To write the value of a single variable to the console use the command:

```
echo variable $<varname>
```

You can also define a variable from an environment variable via the command

```
set env variable $<varname>=<env-var-name>
```

which will substitute the value of the environment variable named `<env-var-name>`. To undefine a variable that is currently defined you may use the following command:

```
unset variable $<varname>
```

Similar to the `set variable` command, all leading/trailing whitespace around the variable name is ignored. Note that calling `unset variable` on a variable that is undefined has no effect. In addition, you may use the following command to undefine all currently defined variables:

```
unset all variables
```

Once a variable is defined, you can obtain its value via variable substitution simply by writing the variable’s name followed directly by a dollar sign ($). For example, the **grok** file commands

```
set variable $path1=D:\projects\my_project\include_files
set variable $path2=D:\projects\my_project\init_files

include $path1$\file1.include

initial head from file
$path2$\head0.txt
```

are equivalent to

```
include D:\projects\my_project\include_files\file1.include

initial head from file
D:\projects\my_project\init_files\head0.txt
```

Note that using the value of an undefined variable will result in a warning message being written to the console.

As discussed above, pre-processor variables are supported by the **grok** file, material properties files, and by all files included via an Include command. It is important to keep in mind that pre-processor variables have global scope among these files. For example, if your **grok** file defines a variable and then includes a file, that variable will be visible within the included file. If the included file then defines a variable with the same name, its value will be overwritten and will persist after the include statement has been processed. The same is true for the material properties files. Therefore, as a best practice, we recommend defining all pre-processor variables at the top of your **grok** file.

We now describe in detail the various actions of the pre-processor, giving instructions for setting up the *prefix*.grok file where necessary.

### 2.1.3 Units and Physical Constants

The units used in the program are not preset, although a default of kilogram-metre-second units is assumed and used to define the values of certain physical constants as discussed below. The user should decide which units will be used for mass (M), length (L), and time (T) for the various input variables, issue the appropriate units instruction (or assign appropriate values for the physical constants) and then consistently use those chosen units for all other input data. The units of temperature ($\Theta$), for example in the case of thermal transport, are expected to be in degrees Celsius unless stated otherwise. For example, if you want to specify the dimensions of your domain in metres and the time at which you want a solution is in seconds, then all measures of length and time will have to be in metres and seconds, respectively. The hydraulic conductivity should therefore be specified in m s$^{-1}$, a pumping rate in m$^3$ s$^{-1}$, etc. The program does not perform any checks to ensure unit consistency.

Default values are assigned for the gravitational acceleration and fluid properties which correspond to standard values in the kilogram-metre-second system. These parameters are used when defining the properties of fractures, open wells and tile drains.

The following default values will be used for the physical constants and correspond to typical values in the kilogram-metre-second system:

- Gravitational acceleration $g = 9.80665$ m s$^{-2}$, Equation 2.3.
- Fluid density $\rho = 1000.0$ kg m$^{-3}$, Equation 2.13.
- Fluid viscosity $\mu = 1.124 \times 10^{-3}$ kg m$^{-1}$ s$^{-1}$, Equation 2.13.
- Fluid compressibility $\alpha_w = 4.4 \times 10^{-10}$ kg$^{-1}$ m s$^2$, Equation 2.15.
- Fluid surface tension $\chi = 0.07183$ kg s$^{-2}$, Equation 1.6.

If you are using different units or you want to change the default values you can do so using the following instructions.

## Units: kilogram-metre-minute

Converts the default values given above into the kilogram-metre-minute system. This instruction also converts the default properties defined in the code for all media types (see Section 2.8.1). Note, however, that it does not convert properties specified in any *prefix*`.grok`, `.mprops`, etc. files. Similar instructions exist for converting to the following systems:

- Kilogram-metre-hour
- Kilogram-metre-day
- Kilogram-metre-year
- Kilogram-centimetre-second
- Kilogram-centimetre-minute
- Kilogram-centimetre-hour
- Kilogram-centimetre-day
- Kilogram-centimetre-year

• • •

You can change the default values of the physical constants using the following instructions. If you change the default units from the kilogram-metre-second system, then you must make sure the new values are in the new system of units.

## Gravitational acceleration

1. **grav** Gravitational acceleration constant [L T$^{-2}$], $g$ in Equation 2.3.

•••

---

### Reference fluid density

1. **rho** Fluid density [M L$^{-3}$], $\rho$ in Equation 2.13.

•••

---

### Reference fluid viscosity

1. **visc** Fluid viscosity [M L$^{-1}$ T$^{-1}$], $\mu$ in Equation 2.13.

•••

---

### Fluid compressibility

1. **wcomp** Fluid compressibility [M$^{-1}$ L T$^{2}$], $\alpha_w$ in Equation 2.15.

•••

---

### Zero fluid compressibility

Assigns a value of zero for fluid compressibility (i.e., incompressible).

•••

---

### Fluid surface tension

1. **tensn** Fluid surface tension [M T$^{-2}$], $\chi$ in Equation 1.6.

•••

## 2.1.4 Array Dimensioning

When reading and allocating default array sizes (Task 1), **grok** first checks for the existence of a file `array_sizes.default` in the directory where the *prefix*`.grok` file is located. If it is not found, the file is automatically created and default array sizes are written, which are then used by the pre-processor. Associated with each default are a descriptor and a default value. A portion of the file is shown here:

```
dual: material zones
        20
dual flow bc: flux nodes
     10000

...etc...

tiles: flux function panels
        20
wells: injection concentration function panels
       100
end
```

So, for example, the default maximum number of dual continuum material zones is 20. If the problem is defined such that an array size exceeds the default maximum (e.g., the number of node sheets in the $z$-direction for layered grids exceeds 50) then **grok** will halt execution and issue an error message (to the screen and the *prefix*`o.eco` file) of the form:

```
*********************************************
*** DIMENSIONING ERROR, HALTING EXECUTION ***
*********************************************

 Pre-processor request exceeds default array size

 mesh: node sheets in z for layered grids
 Default value: 50
 Requested value: 100

 Increase the default value in file array_sizes.default
```

Given the descriptor in the error message, you can now edit the `array_sizes.default` file and increase the appropriate value. When you run **grok** again, it will read the new default value from the file.

**HydroGeoSphere** does not utilize the file `array_sizes.default`, but instead uses exact array sizes determined and passed by **grok**.

Remember, this process is problem dependent, and each time you run **grok** in a different directory, a fresh `array_sizes.default` file will be generated with default values.

### 2.1.5 Utility

The section is intended for general commands that do not belong to any one section of the reference manual.

The following command can be used to add notes or documentation to the *.eco* file as well as to the console while **grok** is processing the *.grok* file. It works by simply echoing a single line of text.

### Print

1. **text** Line of text to be echoed.

Prints the specified line of text to the console and the *.eco* file.

●●●

The following command can be use to read a vertical, end-terminated list from **grok** and then write that list to a user specified ASCII file.

### Write list

1. **filename** Name of the list file.
2. **list_item(i),...end** List to be written to file.

Writes the list, one item per line, to an ASCII file with the specified filename.

●●●

## 2.2 Problem Identification

The first section of the *prefix*`.grok` file should consist of a description of the problem being defined. As for the rest of the file, blank lines and lines beginning with an exclamation point (!) are ignored.

The description can contain from zero up to as many lines as the user requires to describe the problem. Each line can contain up to 60 characters and any lines exceeding this limit will be truncated. The description is printed at the beginning of the listing files for **grok** (*prefix*`o.eco`) and **HydroGeoSphere** (*prefix*`o.lst`). Ideally, the first line of the description should summarize the problem in one clear and concise statement.

The user must signal the end of the description using the End instruction.

### End

This instruction signals the end of the description at which point control is passed back to the pre-processor.

●●●

## 2.3 Grid Generation

The next section of the *prefix*.grok file should consist of instructions for grid generation followed by an End instruction.

Currently, **grok** is capable of generating grids which are composed of either hexahedral blocks or triangular prisms. Figure 2.1 shows the local node numbering conventions for each of these elements and also the positive directions of the $x$-, $y$-, and $z$-axes.

We will first discuss options for generating simple grids, followed by irregular grids.

### 2.3.1 Simple Grids

Simple grids can be generated for rectangular domains which are adequate for many problems. They can have uniform or variable element sizes and can be made of hexahedral block or triangular prismatic elements. Each element in the grid is given a default zone number of 1.

#### Generate uniform blocks

1. **xlen, nbx, (x0)** Domain length [L] and number of blocks in the $x$-direction, the optional origin in the $x$-direction [L] (zero by default).
2. **ylen, nby, (y0)** Domain length [L] and number of blocks in the $y$-direction, the optional origin in the $y$-direction [L] (zero by default).
3. **zlen, nbz, (z0)** Domain length [L] and number of blocks in the $z$-direction, the optional origin in the $z$-direction [L] (zero by default).

Generates a grid for a rectangular domain made up of uniform blocks. In this case, the grid is formed by subdividing the domain in the $x$-direction into **nbx** blocks, each of length **xlen/nbx**. The domain is subdivided in a similar fashion in the $y$- and $z$-directions, using the other input parameters.

•••

#### Generate uniform prisms

Generates a grid for a rectangular domain made up of uniform prisms. Requires identical input to the routine Generate uniform blocks described above. In this case though, instead of generating block elements, this instruction generates prism elements by subdividing each block into two prism elements.

•••

#### Generate variable blocks

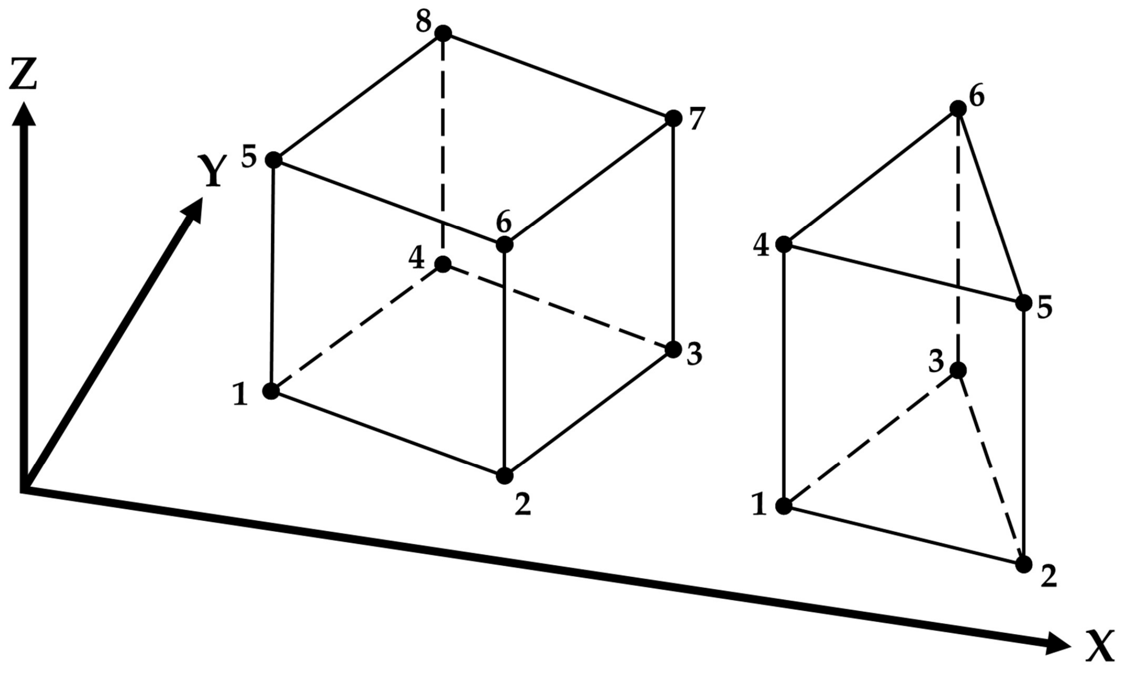


Figure 2.1: Element types and local node numbering conventions.

1. **nx** Number of nodes in the $x$-direction.
2. **x(i), i=1,nx** Nodal $x$-coordinates [L].
3. **ny** Number of nodes in the $y$-direction.
4. **y(i), i=1,ny** Nodal $y$-coordinates [L].
5. **nz** Number of nodes in the $z$-direction.
6. **z(i), i=1,nz** Nodal $z$-coordinates [L].

Generates a grid for a rectangular domain made up of variably-sized blocks. It is almost identical to the Generate uniform blocks instruction except that instead of entering a domain length in each direction we enter a list of coordinates, which are each used to define the position of a plane of nodes along that axis.

•••

## Generate variable prisms

Generates a grid for a rectangular domain made up of variably-sized prisms. Requires identical input to the routine Generate variable blocks described above. In this case though, instead of generating block elements, this instruction generates prism elements by subdividing each block into two prism elements.

●●●

### 2.3.2 Interactive Block Grids

Interactive block instructions can be used to generate a grid made up of variably-sized blocks. The user can grade the mesh as desired in each of the three principal directions. This is particularly useful for regions in which fine meshes are required, for example, near a discrete fracture or well.

Note that these instructions cannot be used in conjunction with the other grid generation instructions such as Generate uniform block, Generate uniform prisms, Generate variable blocks, or Generate variable prisms.

---

#### Generate blocks interactive...End

Causes **grok** to begin reading a group of interactive block instructions until it encounters an End instruction. The group should contain of at least one instruction for each of the principal directions.

●●●

The available instructions are:

---

#### Grade x

1. **x1, x2, dxstart, xfac, dxmax** Starting $x$-coordinate [L], ending $x$-coordinate [L], starting element size, element size multiplication factor, and maximum element size.

Grid lines (i.e. elements) are generated along the $x$-axis from **x1** to **x2** which grade up in size from **dxstart** to **dxmax**. Element sizes are increased steadily by a factor of **xfac**.

●●●

---

#### Grade y

As above but for the $y$-axis.

●●●

---

#### Grade z

As above but for the $z$-axis.

●●●

The instructions used to generate the mesh shown in Figure 2.2 are:

```
generate blocks interactive
grade x
 75.0     0.0   0.01   1.5  5.0
grade x
 75.0   100.0   0.01   1.5  5.0
grade x
125.0   100.0   0.01   1.5  5.0
grade x
125.0   200.0   0.01   1.5  5.0
grade y
100.0     0.0   0.01   1.5  5.0
grade y
100.0   200.0   0.01   1.5  5.0
grade z
  1.0     0.0   0.25   1.0  0.25
grade z
  3.0     1.0   0.01   1.3  0.25
grade z
  3.0    11.0   0.01   1.3  0.25
grade z
 11.0    12.0   0.25   1.0  0.25
end generate blocks interactive
```

### 2.3.3 3-D Random Fracture Generator for Block Grids

The following command can be used to generate a 3-D random fracture network in an orthogonal domain (i.e., composed of 8-node block elements). Fractures with random locations, lengths, and apertures can be generated.

#### Rfgen driver

1. **rfgfile** Name of the file that contains the random fracture grid and fracture generation information.

The structure of the input file is described below.

• • •

#### Grid information

1. **x1, x2** $x$-range [L] of the domain.

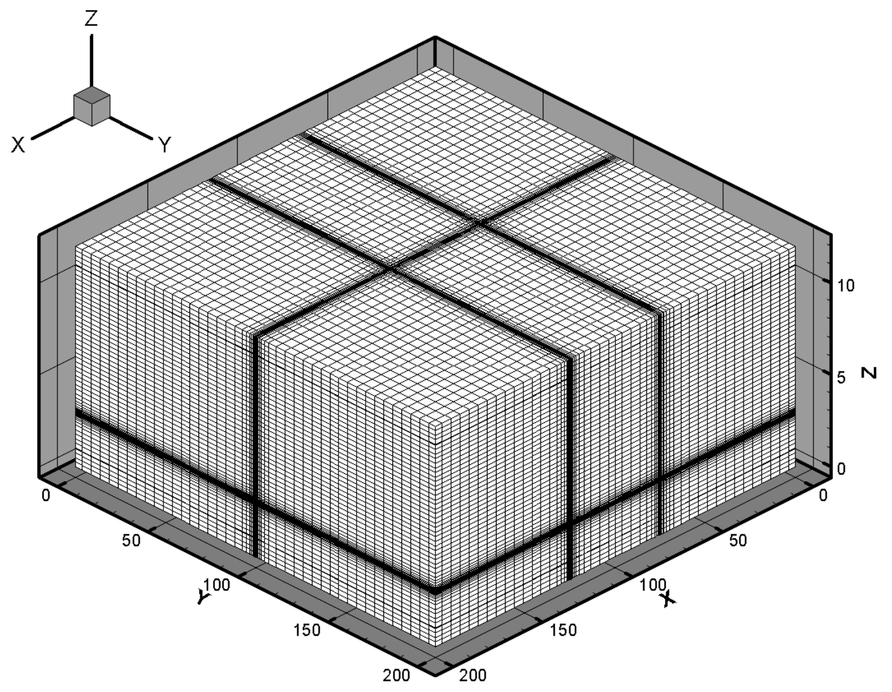


Figure 2.2: Example grid that was created using Generate blocks interactive instructions.

2. **y1, y2** $y$-range [L] of the domain.
3. **z1, z2** $z$-range [L] of the domain.
4. **botfracbnd** Elevation [L] of lowest extent of a fracture. No fractures will be generated below this elevation.
5. **nwell** Number of wells.
6. **xwell(i), ywell(i), i=1,nwell** $xy$-coordinates [L] of the well. Generates $x$- and $y$-grid lines through each point.
7. **xsource1, xsource2** $x$-coordinates [L] of the source. Generates $x$-grid lines at these points.
8. **ysource1, ysource2** $y$-coordinates [L] of the source. Generates $y$-grid lines at these points.
9. **zsource1, zsource2** $z$-coordinates [L] of the source. Generates $z$-grid lines at these points.
10. **mingrspacx, mingrspacy, mingrspacz** Minimum grid spacing [L] in the $x$-, $y$-, and $z$-directions, respectively. For example, a **mingrspacx** value of 1 would ensure that no gridlines are less than 1 length unit apart along the $x$-axis.

11. **fixed_grid** Logical value (T/F) that controls whether grid lines are generated randomly (F) or according to fixed spacing input parameters (T). If true, then read the following:
    (a) **fixed_space** Logical value (T/F) that controls whether uniform (T) or variable (F) grid line spacing is applied. If true, then read the following:
        i. **fixgrspacx, fixgrspacy, fixgrspacz** Fixed spacing [L] in the $x$-, $y$-, and $z$-directions, respectively.

    Otherwise, read the following:
        i. **nx** Number of nodes in the $x$-direction.
        ii. **x(i), i=1,nx** Nodal $x$-coordinates [L].
        iii. **ny** Number of nodes in the $y$-direction.
        iv. **y(i), i=1,ny** Nodal $y$-coordinates [L].
        v. **nz** Number of nodes in the $z$-direction.
        vi. **z(i), i=1,nz** Nodal $z$-coordinates [L].

This instruction should be placed at the top of the file and should not appear more than once.

• • •

## Fracture information

1. **seed** Seed for the random number generator. If this number is changed, a new random number sequence is produced, which in turn causes new realizations of fracture location, length and aperture to be generated.
2. **xmeanfreq** Mean fracture frequency [L$^{-1}$] in the $x$-direction.
3. **ymeanfreq** Mean fracture frequency [L$^{-1}$] in the $y$-direction.
4. **zmeanfreq** Mean fracture frequency [L$^{-1}$] in the $z$-direction.
5. **decay** Aperture decay constant [L$^{-1}$]. Aperture size can be made to decrease with increasing depth. Set to zero for no decay.
6. **lnsbetween** Minimum number of grid lines between fractures.
7. **cap** Maximum number of times to attempt generating a fracture.

This instruction should follow the Grid information instruction and should not appear more than once.

• • •

## Fracture location distribution x-axis

1. **type** An integer value indicating the probability distribution used to generate the variable fracture locations in the $x$-direction. Acceptable values are:

| | |
|---|---|
| 1 | Uniform. |
| 2 | Normal. |
| 3 | Exponential. |

2. **var1, var2** Distribution parameters [L].

For the uniform distribution **var1** is the minimum and **var2** is the maximum. For the normal distribution **var1** is the mean and **var2** is the variance. For the exponential distribution **var1** is the mean and **var2** is the standard deviation.

●●●

The following instructions use the same input data structure as Fracture location distribution x-axis except they are applied to the $y$- and $z$-directions:

Fracture location distribution y-axis
Fracture location distribution z-axis

The following instructions use the same input data structure as Fracture location distribution x-axis to generate fracture lengths in the three principal directions:

Fracture length distribution x-axis
Fracture length distribution y-axis
Fracture length distribution z-axis

The following instructions use the same input data structure as Fracture location distribution x-axis to generate fracture apertures in the three principal orientations:

XY fracture aperture distribution
XZ fracture aperture distribution
YZ fracture aperture distribution

Note that when generating fracture apertures from the normal distribution, random samples are truncated to the interval $(0, \infty)$. A negative fracture aperture generated from either the truncated normal or uniform distribution will result in an error.

The remaining commands are optional but should not be used more than once:

## Vertical fracture from top

1. **vertical_frac_top** Logical value (T/F), which if true, ensures that all vertical fractures start from the top of the domain.

●●●

## Zone fractures how

1. **zone_rfgen_fracs** Controls how fracture zone numbers are assigned. Acceptable values are:

| | |
|---|---|
| 1 | Assign zone numbers by fracture. |
| 2 | Assign zone numbers by orientation. |

If zoned by orientation, horizontal fractures are in zone 1, vertical fractures parallel to the *xy*-axis are in zone 2, and vertical fractures parallel to the *xz*-axis are in zone 3.

●●●

Once the 3-D grid is generated, it is possible to change the random fracture apertures to zoned fracture apertures by following the procedures outlined in Section 2.8.1.4.

### 2.3.4 2-D Random Fracture Generator

The 2-D Random Fracture Generator can be used to generate random fracture networks in two dimensions in an orthogonal domain (i.e., composed of 8-node block elements). Fracture generation is currently restricted to the *xz*-plane, nevertheless, in the *y*-direction, more than one block can be used.

#### Begin 2D random fractures...End

Causes **grok** to begin reading instructions that describe the generation of 2-D random fractures until it encounters an End instruction.

●●●

The following optional instructions can be used to modify the default behaviour of the fracture generator:

#### Number of random fractures

1. **n_rfractures** Number of random fractures to generate.

By default, the 2-D random fracture network will consist of 80 fractures.

●●●

#### Use constant seed

1. **seed** Seed for the 2-D Random Fracture Generator.

Causes the 2-D Random Fracture Generator to use a constant seed (**seed**) to produce the same random fracture network each time **grok** is run.

By default, the 2-D Random Fracture Generator is seeded with a time-dependent value, based on the current system time. In that case, it produces a *different* fracture network each time **grok** is run.

In either case, the seed value is written to the *prefix*`o.eco` file and can be used to generate the same random fracture network many times.

● ● ●

## Generate orientation distribution

1. **or_n_classes** Number of orientation classes.
2. **or_first_class_middle** Middle of the smallest orientation class [deg].
3. **or_last_class_middle** Middle of the largest orientation class [deg].
4. **or_sigma** Standard deviation $\sigma$ [deg] of both Gaussian distributions.
5. **or_mu1** Mean $\mu_1$ [deg] of the first Gaussian distribution.
6. **or_mu2** Mean $\mu_2$ [deg] of the second Gaussian distribution.

Causes **grok** to read the parameters that are used to define the distribution of fracture orientation, which follows a mixed Gaussian distribution according to:

$$P(x) = \frac{1}{2}P_1(x) + \frac{1}{2}P_2(x) \tag{2.1}$$

where the $i$th Gaussian distribution with mean $\mu_i$ and variance $\sigma^2$ has probability density function

$$P_i(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu_i)^2}{2\sigma^2}} \tag{2.2}$$

defined for all $x \in (-\infty, \infty)$.

By default, the values given in Table 2.1 are used to define these relationships.

Table 2.1: Default values for 2-D random fracture orientation.

| Parameter | Value | Unit |
|---|---|---|
| Number of orientation classes | 13 | |
| Middle of the smallest orientation class | 30 | deg |
| Middle of the largest orientation class | 150 | deg |
| Standard deviation of both classes $\sigma$ | 15 | deg |
| Mean of the first Gaussian distribution $\mu_1$ | 60 | deg |
| Mean of the second Gaussian distribution $\mu_2$ | 120 | deg |

• • •

## Generate aperture distribution

1. **ap_n_classes** Number of aperture classes.
2. **ap_first_class_middle** Middle of the smallest aperture class [L].
3. **ap_last_class_middle** Middle of the largest aperture class [L].
4. **ap_lambda** $\lambda$ [$L^{-1}$] of the exponential aperture distribution.

Causes **grok** to read the parameters that are used to define the distribution of fracture aperture, which follows an exponential distribution, according to:

$$P(x) = P_o \cdot e^{-\lambda\, x} \tag{2.3}$$

Note that for a high value of $\lambda$, the exponential distribution becomes steeper and small apertures are more numerous, whereas a small value of $\lambda$ favours larger apertures.

By default, the values given in Table 2.2 are used to define these relationships.

Table 2.2: Default values for 2-D random fracture aperture.

| Parameter | Value | Unit |
|---|---|---|
| Number of aperture classes | 10 | |
| Middle of the smallest aperture class | 50 | microns |
| Middle of the largest aperture class | 300 | microns |
| $\lambda$ of the exponential aperture distribution | 9000 | microns$^{-1}$ |

• • •

## Generate log-normal length distribution

1. **le_n_classes** Number of length classes.
2. **le_first_class_middle** Middle of the smallest length class [L].
3. **le_last_class_middle** Middle of the largest length class [L].
4. **lognormal_m** $\mu$ [-] of the log-normal distribution.
5. **lognormal_s** $\sigma$ [-] of the log-normal length distribution.

Causes **grok** to read the parameters that are used to define the distribution of fracture length, which follows a log-normal distribution according to:

$$P(x) = \frac{1}{\sigma x \sqrt{2\pi}}\, e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{2.4}$$

By default, the values given in Table 2.3 are used to define these relationships.

Table 2.3: Default values for 2-D random fracture length, log-normal distribution.

| Parameter | Value | Unit |
|---|---|---|
| Number of length classes | 10 | |
| Middle of the smallest length class | $0.1 \cdot \min(L_x, L_z)$[†] | m |
| Middle of the largest length class | $\min(L_x, L_z)$ | m |
| $\mu$ of the log-normal distribution | 2.9 | |
| $\sigma$ of the log-normal length distribution | 0.45 | |

[†] The symbols $L_x$ and $L_z$ denote the length of the simulation domain the $x$- and $z$-directions, respectively.

It should be noted that the log-normal distribution $\Lambda(\mu, \sigma^2)$ is obtained from the distribution $\Lambda(0, \sigma^2)$ via:

- Stretching by $e^{\mu}$ in the $x$-direction.
- Stretching by $e^{-\mu}$ in the $z$-direction.

Thus, larger values for $\mu$ will move the peak to the right. Altering the standard deviation $\sigma$ will have an impact on the scattering of the distribution where a small $\sigma$ leads to less scattering and a sharper peak.

• • •

## Exponential length distribution

Causes **grok** to use an exponential distribution of the fracture trace, as opposed to the default log-normal distribution.

• • •

## Generate exponential length distribution

1. **le_n_classes** Number of length classes.

2. **le_first_class_middle** Middle of the smallest length class [L].
3. **le_last_class_middle** Middle of the largest length class [L].
4. **le_lambda** $\lambda$ [$\mathrm{L}^{-1}$] of the exponential length distribution.

Used in conjunction with the Exponential length distribution instruction, this causes **grok** to read the parameters that are used to define the distribution of fracture length, which follows an exponential distribution according to:

$$P(x) = P_o \cdot e^{-\lambda\, x} \tag{2.5}$$

Note that for a high value for $\lambda$, the exponential distribution becomes steeper and short fractures are more numerous whereas a small $\lambda$ favors longer fractures.

By default, the values given in Table 2.4 are used to define these relationships.

Table 2.4: Default values for 2-D random fracture length, exponential distribution.

| Parameter | Value | Unit |
|---|---|---|
| Number of length classes | 10 | |
| Middle of the smallest length class | $0.1 \cdot \min(L_x, L_z)$[†] | m |
| Middle of the largest length class | $\min(L_x, L_z)$ | m |
| $\lambda$ of the exponential length distribution | 0.05 | $\mathrm{m}^{-1}$ |

† The symbols $L_x$ and $L_z$ denote the length of the simulation domain in the $x$- and $z$-directions, respectively.

●●●

## Output random apertures

Writes the generated aperture distribution data and individual fracture apertures to the output file *prefix*`o.rfrac.apertures`.

●●●

## Output random lengths

Writes the generated length distribution data and individual fracture lengths to the output file *prefix*`o.rfrac.lengths`.

●●●

## Output random orientations

Writes the generated orientation distribution data and individual fracture orientations to the output file *prefix*`o.rfrac.orientations`.

●●●

## Output random fractures

Writes fracture zone aperture, conductivity and location data to the output file *prefix*`o.rfrac.fractures`.

●●●

Figure 2.3 gives an overview of the default distributions which the 2-D Random Fracture Generator employs. The orientation distribution (Figure 2.3a) is based on the assumption that tectonic stress results in the creation of two fracture families as depicted in Figure 2.3b. However, upon assigning identical values for $\mu_1$ and $\mu_2$, the distribution collapses to a one-peak distribution. The default distribution for the aperture (Figure 2.3c) is exponential and can be modified by the user. By default, the fracture traces are distributed log-normally (Figure 2.3d), which can be changed to exponential. Note that the fracture trace distribution depends on the domain dimensions. Here, a block has been used with $L_x = 100$ m, $L_y = 1$ m, and $L_z = 50$ m.

The following instructions were used to generate the irregular fracture network shown in Figure 2.4. Note the dominance of the two orientations 80° and 135°.

```
!_______________________ grid definition

generate uniform blocks
100.0 200
1.0 1
50.0 100

adapt grid to fractures
3
end

...etc...

!_______________________ fracture media properties

use domain type
fracture
```

```
properties file
eval.fprops

begin random fractures

use constant seed
0.5

number of random fractures
70

exponential length distribution

generate orientation distribution
10
60.
150.
10.
80.
135.

output random apertures
output random lengths
output random orientations
output random fractures

end

read properties
fracture
```

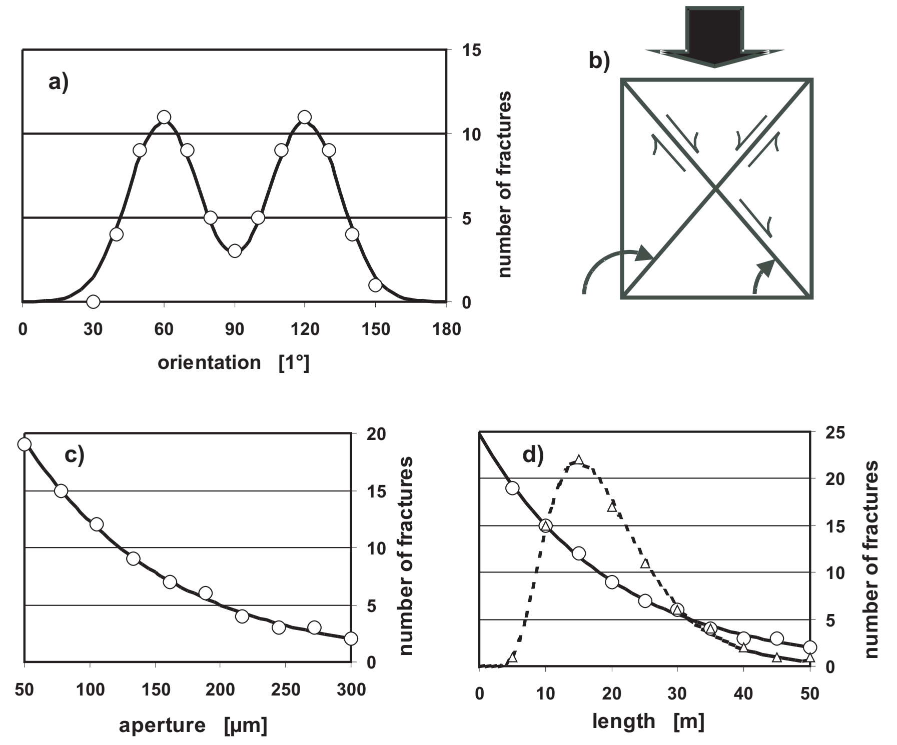


Figure 2.3: Default random fracture distribution.

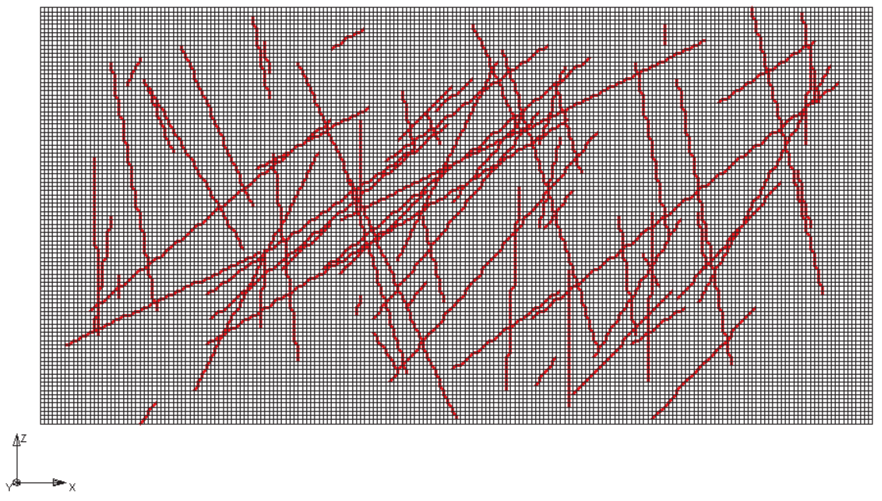


Figure 2.4: Example of an irregular fracture network.

### 2.3.5 Interactive 3-D Mesh Generator

Irregular grids can be generated by supplying nodal coordinates, element incidences and element zones for a 2-D slice which is composed of triangular or quadrilateral elements. Currently, triangles and quadrilaterals can not be mixed in the same slice. These slices can then be replicated to form a 3-D mesh composed of 6-node prisms (from triangles) or 8-node hexahedra (from quadrilaterals).

#### 2.3.5.1 Defining a 2-D Mesh

The following instructions can be used to obtain 2-D slice data.

---

##### Generate uniform rectangles

1. **xlen, nbx, (x0)** Domain length [L] and number of rectangles in the $x$-direction, the optional origin in the $x$-direction [L] (zero by default).
2. **ylen, nby, (y0)** Domain length [L] and number of rectangles in the $y$-direction, the optional origin in the $y$-direction [L] (zero by default).

Generates a 2-D grid for a rectangular domain made up of uniform rectangles. Each rectangular element will be assigned a default zone number of 1. It is identical to the Generate uniform blocks instruction except that we drop the $z$-axis parameters.

•••

---

##### Generate variable rectangles

1. **nx** Number of nodes in the $x$-direction.
2. **x(i), i=1,nx** Nodal $x$-coordinates [L].
3. **ny** Number of nodes in the $y$-direction.
4. **y(i), i=1,ny** Nodal $y$-coordinates [L].

Generates a 2-D grid for a rectangular domain made up of variably-sized rectangles. Each rectangular element will be assigned a zone number of 1. This command is almost identical to the Generate variable blocks instruction except that we drop the $z$-axis parameters.

•••

---

##### Generate rectangles interactive

This instruction works in exactly the same way as the Generate blocks interactive instruction described in Section 2.3.2, except that input is limited to the $x$- and $y$-directions and a 2-D mesh of 4-node rectangular elements is generated.

●●●

## Generate uniform triangles

1. **xlen, nbx, (x0)** Domain length [L] and number of rectangles in the $x$-direction, the optional origin in the $x$-direction [L] (zero by default).
2. **ylen, nby, (y0)** Domain length [L] and number of rectangles in the $y$-direction, the optional origin in the $y$-direction [L] (zero by default).

Generates a 2-D grid for a rectangular domain made up of uniform triangles. Each triangular element is assigned a default zone number of one. This command is identical to the command Generate uniform prisms except that we drop the $z$-dependence.

●●●

## Read gms 2d grid

1. **gmsfile** Filename of the 2-D GMS formatted mesh.

Generates a 2-D grid from the mesh defined by the input file. The format of this file is described in detail in Appendix F.1 and is compatible with that produced by the Groundwater Modeling System (GMS) software.

●●●

## Read algomesh 2d grid

1. **ah2_mesh** Filename of the 2-D `.ah2` mesh exported from AlgoMesh.

Generates a 2-D grid from the mesh defined by the input file.

●●●

## Read vtk 2d grid

1. **vtk_file** Filename of the 2-D legacy VTK file.

Generates a 2-D grid from the mesh defined by a legacy VTK file. The input file must use the ASCII file type and either the structured point, structured grid, rectilinear grid, or unstructured grid dataset type. For the structured point, structured grid, or rectilinear grid dataset types, the 2-D grid is defined from the mesh $xy$-coordinates with the $k$-index ($z$-coordinate index) equal to zero. For the unstructured grid dataset type, the mesh must consist of a single cell type, either VTK_TRIANGLE or VTK_QUAD.

●●●

## Refine 2d grid

Causes **grok** to refine 2-D irregular triangular grid (one triangle to four triangles). This command can be repeated multiple times after the 2-D grid has been imported. Note that the number of nodes increases by about four times with this command.

●●●

## Reduce 2d grid, boundary file

1. **polygon_file** Filename of a text file that defines a polygon by the $x$- and $y$-coordinates of its vertices, one vertex per line of the file. Note that the number of lines in this file must be the same as the number of vertices in the polygon and the first and last vertices must be identical.

This command clips an existing 2-D grid to those elements/nodes that belong to the input polygon, renumbering nodes and elements in the process. A 2-D element is defined to belong to a polygon when all its vertices belong to that polygon. Combined with the command Refine 2d grid, this command provides a basis for telescopic mesh refinement.

●●●

For a 2-D slice made of 4-node rectangular elements, the following instructions can be used to remove elements:

## Remove rectangles with shapefile

1. **filename** Name of the shapefile without the file extension.
2. **unproject_file** Logical value (T/F), which if true, causes **grok** to read grid unprojection data and apply it to the data read from the shapefile. Note: this option is no longer supported an should always be set to false (F).
3. **project_file** Logical value (T/F), which if true, causes **grok** to read grid projection data and apply it to the data read from the shapefile. Note: this option is no longer supported an should always be set to false (F).
4. **outside** Logical value (T/F), which if true, causes elements located outside the area defined in the shapefile to be removed. Otherwise, elements located inside the area are removed.

Update an existing 2-D grid by removing elements based on whether they are either inside or outside a polygon defined by the shapefile. Inclusion/exclusion of an element is based on its centroid $(x_c, y_c)$. Currently, this command applies only to rectangular elements.

•••

## Remove rectangles with blanking file

1. **filename** Filename of the rectangle blanking file that defines a single polygon in the Golden Software Blanking File (`.bln`) format described here.

Update an existing 2-D grid by removing elements based on whether they are either inside or outside a polygon defined by the blanking file. Inclusion/exclusion of an element is based on its centroid $(x_c, y_c)$. Currently, this command applies only to rectangular elements.

•••

#### 2.3.5.2 3-D Mesh Generation

Once you have a 2-D slice, you have the option of exiting the grid definition procedure, which will cause **grok** to automatically generate a unit thickness 3-D grid. It does this by duplicating the 2-D slice and constructing the appropriate 6-node prism or 8-node hexahedral element incidences and assigning a unit element length perpendicular to the slice. The element zone numbers for the slice are used to assign default zone numbers for each element. Such a grid could be used to simulate 2-D cross-sectional problems.

More often, you will want to generate a 3-D layered grid, perhaps with topography defined by a DEM (Digital Elevation Model) and/or uneven layer contacts based on the observed hydrostratigraphy.

To do so you should start by issuing the following instruction:

## Generate layers interactive...End

Causes **grok** to begin reading a group of 3-D grid generation instructions until it encounters an End instruction.

•••

The basic procedure is to build up the 3-D mesh by defining the base, then adding layers one at a time from the base to ground surface. By default, the domain will contain a single layer, one element high with a base elevation of zero and a top elevation of 1, and the element zone numbering scheme from the 2-D slice will be used to assign the 3-D mesh element zone numbers. Instructions that change the default behaviour are described below. These commands are optional, apply globally, and need only be issued once at the start of grid generation.

## Zone by layer

Causes **grok** to assign the 3-D mesh layer number to the element zone number. By default, the element zone numbering scheme from the 2-D slice is used to assign the 3-D mesh element zone number. Note that this command should be issued before any New layer...End commands.

• • •

## Minimum layer thickness with fixed top elevation

1. **min_thick** Minimum thickness [L].

Causes **grok** to enforce a minimum thickness constraint for all layers. At nodes where the computed layer top elevation is less than or equal to the current base elevation, **min_thick** is subtracted from the top elevation to get the base elevation. In contrast to the command Minimum layer thickness, this command applies to all layers and adjusts layers from the top down, maintaining the original surface layer elevations. If this constraint is not enforced, then **grok** will stop and issue a warning message if the computed top elevation is less than or equal to the current base elevation. Note that this command should be issued before any New layer...End commands.

• • •

## Base elevation...End

Causes **grok** to begin reading a group of base elevation instructions until it encounters an End instruction. Available instructions are described in Section 2.3.5.4. By default, the base elevation of the domain will be set to zero.

• • •

#### 2.3.5.3 Adding a New Layer

For each layer in the domain, you should include a single instance of the following instruction:

## New layer...End

Causes **grok** to begin reading a group of new layer instructions until it encounters an End instruction.

By default, the top elevation of the layer will be set to zero by assigning all nodes in the lowermost 2-D slice to have a $z$-coordinate value of zero. You can change the layer top elevation using the instructions described in Section 2.3.5.4.

• • •

The following instructions are optional:

---

## Layer name

1. **layer_name** Layer name.

Changes the layer name, which defaults to "Layer *n*", where *n* is the current layer number.

●●●

---

## Minimum layer thickness

1. **min_thick** Minimum thickness [L].

Causes **grok** to enforce a minimum thickness constraint for the current layer. At nodes where the computed layer top elevation is less than or equal to the current base elevation, **min_thick** is added to the current base elevation to get the top elevation. If this constraint is not enforced, **grok** will stop and issue a warning message if the computed top elevation is less than or equal to the current base elevation.

●●●

By default, a new layer will not be subdivided vertically unless one of the following two instructions is issued:

---

## Uniform sublayering

1. **nsublayer** Number of sublayers.

Divides the layer vertically into **nsublayer** elements, which will each have the same height, equal to the top elevation minus the current base elevation divided by **nsublayer**.

●●●

---

## Proportional sublayering

1. **nsublayer** Number of proportional sublayers.
2. **sub_thick(i), i=1,nsublayer** Proportional thicknesses [-] in order from top to bottom.

This instruction can be used if you want to refine the mesh vertically, for example in the subsurface near the surface flow domain or a fracture. It is important to understand that the variable **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 would subdivide the current layer vertically into three elements, between the current base and top elevation, with element height proportions of 0.1, 1, and 10 from top to bottom:

```
proportional sublayering
    3
    0.1
    1.0
    10.0
```

• • •

## Offset top

1. **thickness** Thickness [L] by which to offset the layer top elevation.

Causes the elevation of the top layer to be offset vertically by the given value. This command can be used to create a surface at a given distance below another surface.

• • •

## Offset base

1. **thickness** Thickness value [L] by which to offset the base elevation.

Causes the base elevation to be offset vertically by the given value. For example, these instructions create two layers with the elevations 2 and 1 metres below the elevation defined in the raster `gs.asc` and the raster elevation:

```
base elevation
    elevation from raster
    gs.asc

    offset base
    -2.0
end

new layer
    uniform sublayering
    10

    elevation from raster
```

```
        gs.asc

        offset top
        -1.0
    end

    new layer
        uniform sublayering
        3

        elevation from raster
        gs.asc
    end
```

●●●

#### 2.3.5.4 Elevation Instructions

These instructions are used to define 3-D mesh base elevations and new layer top elevations.

## Elevation constant

1. **elev** Elevation value [L].

Assigns a constant elevation of **elev** to all nodes in the surface mesh.

●●●

## Elevation from file

1. **filename** Filename of the ASCII file containing the elevation [L] values.

This command assigns an elevation to each node in the surface mesh from the input file. The file should contain one elevation value per line for each node in the surface mesh. The order of nodal elevations in the input file must match the node order within a 2-D mesh sheet.

●●●

## Elevation from raster

1. **rasterfile, (band)** Name of raster file that contains base elevation [L] values and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).

This command assigns an elevation to each node in the surface mesh by interpolating them via bilinear interpolation from the specified raster band.

●●●

### Elevation from tsurf file

1. **filename** Filename of GoCAD tsurf file. Note that the length units in this file must match the model’s length units.

This command assigns elevations by interpolating to a surface consisting of triangular elements. The surface is stored as a GoCAD tsurf file. Note that this command applies only to triangular prism or hexahedral block meshes.

●●●

### Elevation from xz pairs

1. **xval(i), zval(i)...end** $xz$-coordinates [L] list.

Listed $xz$-coordinate pairs should be given in order from the smallest to largest $x$-value. For each node in the 2-D grid, the $x$-coordinate of the node is used to determine its position in the list, and a $z$-coordinate is then interpolated from the neighbouring $xz$-pairs.

●●●

## 2.3.6 Reading an Existing 3-D Grid

In some cases, the grid generation step can be very time consuming. If this is so, then the following instruction can be used to read a grid which was generated in a previous run:

### Read 3D grid

The grid whose prefix matches that of the *prefix*`.grok` file will be read in. For example, these instructions show how to set up the grid generation section of the *prefix*`.grok` file to use the Read 3d grid instruction:

```
    ! generate the grid for the first run
    read algomesh 2d grid
```

```
mesh_slice.ah2

generate layers interactive
    new layer
        elevation from raster
        layer1.asc
    end

    new layer
        elevation from raster
        layer2.asc
    end
end

skip on
! read previously defined grid for subsequent runs
read 3D grid
skip off
end
```

In the first run, one can read a slice and generate the layered grid. It is important that the instruction Write faces and segments be included if you are using any of the Choose face or Choose segment instructions later in the *prefix*.`grok` file. In subsequent runs, one can skip over the grid generation commands and use the Read 3D grid instruction instead.

●●●

## Read 3D grid, ascii

Causes **grok** to read the ASCII files *prefix*.`coordinates` and *prefix*.`elements` to create a 3-D mesh. File formats are as follows:

*prefix*.`coordinates`

1. **nn** Total number of nodes.
2. **x(i), y(i), z(i), i=1,nn** Nodal *xyz*-coordinates [L].
3. **nx, ny, nz, nsptot** Number of grid lines in the *x*-, *y*-, and *z*-coordinates, and the number of species for transport. Note that for an unstructured mesh **nx** and **ny** are ignored; **nsptot** = 0 always.
4. **ne2d** Number of triangles or rectangles in a 2-D node sheet.
5. **tetramesh** Logical value (T/F) always set to false.
6. **gal_mode** Logical value (T/F) always set to false.
7. **write_face_seg** Logical value (T/F) always set to false.

8. **nb2d** Maximum number of nodes connected to a node for a 2-D triangular mesh.

*prefix*`.elements`

1. **nln** Number of nodes within a single element (6: triangular prism, 8: hexahedron).
2. **ne** Total number of elements.
3. **inc(i, j) i=1,nln, j=1,ne** Node numbers for element incidences for each element.
4. **zone(i), i=1,ne** Element ID (zone number) for each element.

●●●

### 2.3.7 Manipulating the 3-D Grid

#### Adapt grid to fractures

1. **adapt_g2f_mode** An integer value indicating how the grid is adapted to inclined fractures.

If block elements are used, two inclined fractures may intersect in the middle of an element instead of on a grid node, so the fractures will not be connected unless additional nodes are specified.

Acceptable values for the variable **adapt_g2f_mode** and the actions taken in each case are:

| | |
|---|---|
| 0 | No action is taken. |
| 1 | New grid lines are added. |
| 2 | The block element is substituted by four prisms. |
| 3 | Inclined faces are not selected in the block element where the problem occurs. |

The default value is 1.

●●●

#### Adjust absolute nodal elevation by sheet

1. **min_thick** Minimum thickness [L].
2. **node(i), sheet(i), elevation(i)...end** Target node id, sheet number, and absolute elevation [L].

Sets the elevation of the target nodes to the specified elevation in the list. Each line of the node list specifies the 2-D node id, sheet number, and target absolute elevation. This instruction adjusts layer thickness to comply with the defined minimum thickness. Please note that the nodes defined earlier in the list have less priority than the nodes defined later, meaning that the elevation of first nodes defined in the list might be affected by the elevation and layer thickness adjustments applied to the last nodes.

●●●

### Adjust relative nodal elevation by sheet

1. **min_thick** Minimum thickness [L].
2. **node(i), sheet(i), elevation(i)...end** Target node id, sheet number, and relative elevation [L].

Sets the elevation of the target nodes to the specified elevation in the list. Each line of the node list specifies the 2-D node id, sheet number, and target relative elevation. This instruction adjusts layers thickness to comply with the defined minimum thickness. Please note that the nodes defined earlier in the list have less priority than the nodes defined later, meaning that the elevation of first nodes defined in the list might be affected by the elevation and layer thickness adjustments applied to the last nodes.

●●●

### 2.3.8 Ending Grid Generation

### End

Signals the end of the user-controlled portion of the grid definition section of the input data file. At this stage, the pre-processor will automatically perform grid modifications if appropriate. For example, if you read in 2-D slice data but did not specify layer information using for example, the Generate layers interactive...End instruction, the pre-processor would generate a default 3-D system by duplicating the 2-D slice to form a single layer of unit-thickness elements.

●●●

## 2.4 Selecting Mesh Components

In order to assign boundary conditions, material properties etc. we need to be able to choose subsets of the grid. The method of choice must be flexible and easy to use as well as being able to handle complex input requirements.

The following is a list of grid components, ranked in order of increasing complexity:

1. Nodes: used to assign initial heads and first-type boundary conditions.
2. Segments: used to represent wells, tile drains, or observation wells.
3. Faces (triangles or rectangles): used to represent fractures or high-conductivity planes (as 2-D triangular or rectangular elements) and to assign second- and third-type boundary conditions to these as well as 3-D prism or block elements.
4. Elements (blocks or prisms): sometimes used to assign hydraulic conductivities or distribution coefficients.
5. Zones: generally used to assign material properties such as hydraulic conductivity. Elements are grouped into zones by assigning them the same ID number.

We will assign to all members of a grid component an attribute called 'chosen' that can be toggled on or off by the user. If an attribute is chosen for certain members of a component, then subsequent instructions issued by the user will affect those members only. For example, the following section of a hypothetical *prefix*`.grok` file would initially turn off all chosen nodes (i.e. instruction Clear chosen nodes which requires no further input), then turn on only those nodes satisfying the requirement that they are within $10^{-5}$ distance units of the plane defined by the equation $x = 0$ (i.e. instruction Choose nodes x plane followed by two lines of input).

```
clear chosen nodes
choose nodes x plane
0.0                  ! x-coordinate of plane
1.e-5                ! distance criteria
```

Once these nodes were chosen, we could set the property of interest by issuing another set of instructions, for example:

```
create node set
my_node_set

boundary condition
  type
  head

  node set
  my_node_set

  time value table
  0.0 10.0
  end
end
```

In this case we are assigning a constant head of 10.0 to all chosen nodes at time zero, which will apply for the duration of the simulation. We note that the head boundary condition instruction is acting on nodes via the Node set instruction. In general, it is up to the user to be aware of which components each group of instructions acts upon.

The effect of issuing two such instructions in succession is cumulative. For example, the following input would choose nodes that are within $10^{-5}$ distance units of the planes $x = 0$ and $x = 10$.

```
clear chosen nodes

choose nodes x plane
0.0                  ! x-coordinate of plane
1.e-5                ! distance criteria

choose nodes x plane
10.0                 ! x-coordinate of plane
1.e-5                ! distance criteria
```

The following sections introduce all instructions that are available for choosing subsets of the various grid components.

### 2.4.1 Selecting Nodes

The following instructions may be used to alter the set of chosen nodes.

#### Clear chosen nodes

All nodes in the domain will be flagged as *not* chosen. This command is recommended if you are unsure of which nodes are chosen due to previously issued commands.

●●●

#### Clear chosen nodes am

1. **filename** Name of the AlgoMesh chosen nodes file *am_prefix*`.nchos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Nodes flagged as true in the file that are between the top and bottom sheets (inclusive) are flagged as *not* chosen.

●●●

## Clear chosen nodes by shp

1. **filename** Name of the shapefile without the file extension.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Nodes that lie within a polygon defined by the shapefile and between the top and bottom sheets (inclusive) are flagged as *not* chosen. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.*

●●●

## Invert chosen nodes

Inverts chosen nodes. Chosen nodes will be flagged as *not* chosen and the originally unchosen nodes will be chosen.

●●●

## Choose nodes all

All nodes in the domain will be chosen. This command is useful if you wish to assign a property to all nodes in the grid. For example, you could issue this command and then assign a uniform initial head for the problem.

●●●

## Choose node

1. **x1, y1, z1** $xyz$-coordinate [L].

The node closest to the given coordinate will be chosen.

●●●

## Choose node number

1. **node_number** Node number.

The node with number **node_number** will be chosen. You should use this instruction with caution since node numbering will change if the grid structure changes.

●●●

## Choose node xy sheet

1. **x, y, sheet** *xy*-coordinate [L] and sheet number, respectively.

The node nearest to the given coordinates in the specified sheet will be chosen. You should use this instruction with caution since the sheet numbering may change if the grid structure changes.

●●●

## Choose node xy top

1. **x, y** *xy*-coordinate [L].

The node nearest to the given coordinates in the topmost nodesheet sheet will be chosen.

●●●

## Choose nodes x plane

1. **x1** $x$-coordinate [L] of the plane.
2. **ptol** Distance [L] from the plane.

Nodes within distance **ptol** of the plane defined by the equation $x = \mathbf{x1}$ will be chosen. This command is particularly useful when assigning boundary conditions to a specific side of a rectangular domain.

●●●

## Choose nodes y plane

As above but for the $y$-plane.

●●●

## Choose nodes z plane

As above but for the $z$-plane.

●●●

## Choose nodes 3pt plane

1. **x1, y1, z1** *xyz*-coordinates [L] of the first point.

2. **x2, y2, z2** *xyz*-coordinates [L] of the second point.
3. **x3, y3, z3** *xyz*-coordinates [L] of the third point.
4. **ptol** Distance [L] from the plane.

Nodes within distance **ptol** of the plane defined by the three points will be chosen. This command allows you to choose planes of nodes with an arbitrary orientation.

●●●

## Choose nodes 3pt plane bounded

1. **x1, y1, z1** *xyz*-coordinates [L] of the first point.
2. **x2, y2, z2** *xyz*-coordinates [L] of the second point.
3. **x3, y3, z3** *xyz*-coordinates [L] of the third point.
4. **ptol** Distance [L] from the plane.
5. **x4, x5** $x$-range [L] of the block.
6. **y4, y5** $y$-range [L] of the block.
7. **z4, z5** $z$-range [L] of the block.

Nodes within distance **ptol** of the plane defined by the three points and within the rectangular block defined by the three ranges will be chosen.

●●●

## Choose nodes block

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.

Nodes within the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

●●●

## Choose nodes top

All nodes in the top sheet of the domain are chosen.

●●●

## Choose nodes top block

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.

Nodes in the top sheet within the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

●●●

## Choose nodes bottom

All nodes in the bottom sheet of the domain are chosen.

●●●

## Choose nodes sheet

1. **nsheet** Sheet number.

All nodes in this sheet are chosen.

●●●

## Choose nodes top am

1. **filename** Name of the AlgoMesh chosen nodes file *am_prefix*`.nchos.`*description*.

Nodes flagged as true in the file and that are in the top sheet are chosen.

●●●

## Choose nodes am

1. **filename** Name of the AlgoMesh chosen nodes file *am_prefix*`.nchos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Nodes flagged as true in the file that are between the top and bottom sheets (inclusive) are chosen.

• • •

## Choose nodes shp

1. **filename** Name of the shapefile without the file extension.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Nodes that lie within a polygon defined by the shapefile and between the top and bottom sheets (inclusive) are chosen. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.*

• • •

## Choose nodes list

1. **filename** Name of the file that contains the list of node numbers.

Nodes listed in the file are chosen. The input file consists of a list of node numbers, one entry per line. The procedure exits automatically when end-of-file is reached.

• • •

## Choose nodes xyz list

1. **filename** Name of the file that contains the list of $xyz$-coordinates [L].

For each $xyz$-coordinate, the node nearest to that point is chosen. The input file consists of a list of $xyz$-coordinates, one entry per line.

• • •

## Choose nodes xy sheet list

1. **x(i), y(i), sheet(i)...end** $xy$-coordinates [L] and sheet number list.

For each $xy$-coordinate and sheet number in the list, the node nearest to that point is chosen.

• • •

## Choose nodes xy top list

1. **x(i), y(i)...end** *xy*-coordinates [L] list.

For each *xy*-coordinate in the list, the node nearest to that point in the topmost sheet is chosen.

●●●

## Choose surface flow nodes

Nodes currently flagged as surface flow nodes are chosen.

●●●

## Choose nodes top boundary

Nodes around the top edge of the surface flow domain are chosen.

●●●

## Choose nodes lateral boundaries

Selects all nodes belonging to the lateral boundaries of the porous media domain. It is equivalent to choosing all nodes belonging to the boundary edges within each sheet of the mesh.

●●●

## Choose nodes tecplot geometry

1. **geofile** Name of the Tecplot geometry file which contains polyline data.

The node nearest to each point on the polyline is chosen.

●●●

## Choose nodes horizontal circle

1. **x_mid, y_mid, z_mid** *xy*-coordinates [L] of the centre of the circle and elevation [L] of the circle.

2. **radius** Radius [L] of the circle.

3. **ptol** Vertical tolerance [L].

4. **rtol** Horizontal tolerance [L].

Nodes within a vertical distance **ptol** of elevation **z_mid**, and within a horizontal distance **rtol** of the circle with centre **x_mid, y_mid** and radius **radius** are chosen. This command allows you to choose nodes in a domain that has a circular ground-plan.

● ● ●

## Choose nodes active/inactive boundary

Nodes that lie on the boundary between active and inactive elements are chosen. This instruction is intended to be used to select the surface of nodes on top of the active elements so that flow boundary conditions (e.g., head equals elevation) can be assigned.

● ● ●

## Choose nodes 2d-list sheet

1. **nsheet** Sheet number.
2. **filename** Name of the file that contains the list of 2-D node numbers.

The 3-D nodes in the sheet are chosen based on the 2-D node number and the sheet number. If the 2-D node number in the list is larger than the number of nodes in a single mesh sheet but less than or equal to the total number of nodes in the mesh, then it is automatically converted to a corresponding 2-D node number. The input file consists of a list of 2-D node numbers, one entry per line. The procedure exits automatically when end-of-file is reached.

● ● ●

## Choose nodes top 2d-list

1. **filename** Name of the file that contains the list of 2-D node numbers.

The 3-D nodes in the top sheet are chosen based on the 2-D node number. If the 2-D node number in the list is larger than the number of nodes in a single mesh sheet but less than or equal to the total number of nodes in the mesh, then it is converted to a corresponding 2-D node number. The input file consists of a list of 2-D node numbers, one entry per line. The procedure exits automatically when end-of-file is reached.

● ● ●

## Choose nodes polyline by sheet

1. **bot_sheet, top_sheet** Bottom and top sheet numbers (inclusive), respectively.

2. **npts** Number of points defining the polyline.
3. **x(i), y(i), i=1,npts** *xy*-coordinates [L] of points on the polyline within a single sheet. The points should be ordered from one end of the polyline to the other.

Within each sheet between **bot_sheet** and **top_sheet**, nodes that fall on or close to the polyline are chosen.

• • •

## Choose nodes polyline by sheet shp

1. **bot_sheet, top_sheet** Bottom and top sheet numbers (inclusive), respectively.
2. **filename** Name of the shapefile without the file extension.

Within each sheet between **bot_sheet** and **top_sheet**, nodes that fall on or close to the shapefile polyline are chosen. PolyLine, PolyLineM, and PolyLineZ ($z$-coordinate ignored) shape types are supported.

• • •

## Choose nodes polyline top sheet

1. **npts** Number of points defining the polyline.
2. **x(i), y(i), i=1,npts** *xy*-coordinates [L] of points on the polyline in the top sheet. The points should be ordered from one end of the polyline to the other.

Within the top sheet, nodes that fall on or close to the polyline are chosen.

• • •

## Choose nodes polyline top sheet shp

1. **filename** Name of the shapefile without the file extension.

Within the top sheet, nodes that fall on or close to the shapefile polyline are chosen. PolyLine, PolyLineM, and PolyLineZ ($z$-coordinate ignored) shape types are supported.

• • •

## Choose nodes from chosen faces

Nodes that belong to the current set of chosen faces are chosen.

●●●

## Choose nodes from chosen segments

Nodes that belong to the current set of chosen segments are chosen.

●●●

## Choose nodes by zone

1. **zone_number** Zone number.

Nodes that belong to the specified zone are chosen.

●●●

## Choose nodes between zones

1. **zone_number1, zone_number2** Zone numbers.

The interface nodes between two zones are chosen.

●●●

## Write chosen nodes

1. **filename** Filename of output file.

Writes the node number of each chosen node to the specified file, one entry per line.

●●●

## Write chosen nodes xyz

1. **filename** Filename of output file.

Writes the node number and *xyz*-coordinates [L] of each chosen node to the specified file, one entry per line.

●●●

## Write ordered nodes from polyline by sheet

1. **shp_filename** Filename of polyline shapefile (`.shp`).
2. **csv_filename** Filename of CSV output file.
3. **sheet** Node sheet number.

This command generates a list of nodes that belong to the specified node sheet that follow closely to the polylines defined by a shapefile. For each polyline segment, nodes are written in the order of their connectivity to a CSV output file that records the line id (unique identifier for each polyline segment), mesh node number, and nodal *xyz*-coordinates.

●●●

The following commands may be used to generate files that are compatible as input to the boundary condition command Time file table, where it is expected that the boundary condition employs a node selection. These commands are intended to be used in conjunction with the commands: Choose nodes xyz list, Choose nodes xy sheet list, Choose nodes xy top list, to define the boundary condition node selection.

## Make time file table input xyz

1. **filename** Filename of output file.
2. **x(i), y(i), z(i), vals(i)...end** Nodal *xyz*-coordinates [L] and space delimited values (one or more per line) list.

Writes an ASCII file with the specified filename that is compatible with the boundary condition command Time file table. Each line in the input list of the form $(x, y, z, v_1, \ldots, v_n)$ is converted to a line in the output file of the form $(v_1, \ldots, v_n)$ that is associated with the mesh node that is nearest to the point $(x, y, z)$. Lines in the output file occur in increasing order of their corresponding node number, with only unique node numbers retained.

●●●

## Make time file table input xy sheet

1. **filename** Filename of output file.
2. **x(i), y(i), sheet(i), vals(i)...end** Nodal *xy*-coordinates [L], sheet number, and space delimited values (one or more per line) list.

Writes an ASCII file with the specified filename that is compatible with the boundary condition command Time file table. Each line in the input list of the form $(x, y, s, v_1, \ldots, v_n)$ is converted to a line in the output file of the form $(v_1, \ldots, v_n)$ that is associated with the mesh node that is nearest to the point $(x, y)$ in sheet $s$. Lines in the output file occur in increasing order of their corresponding node number, with only unique node numbers retained.

●●●

---

### Make time file table input xy top

1. **filename** Filename of output file.
2. **x(i), y(i), vals(i)...end** Nodal *xy*-coordinates [L] and space delimited values (one or more per line) list.

Writes an ASCII file with the specified filename that is compatible with the boundary condition command Time file table. Each line in the input list of the form $(x, y, v_1, \ldots, v_n)$ is converted to a line in the output file of the form $(v_1, \ldots, v_n)$ that is associated with the mesh node that is nearest to the point $(x, y)$ in the topmost sheet. Lines in the output file occur in increasing order of their corresponding node number, with only unique node numbers retained.

●●●

### 2.4.2 Selecting Segments

The following instructions may be used to alter the set of chosen segments.

---

### Clear chosen segments

All segments in the domain are flagged as *not* chosen. This command is recommended if you are unsure of which segments are chosen due to previously issued commands.

●●●

---

### Choose segments all

All segments in the domain are flagged as chosen. This command is useful if you wish to assign a property to all segments in the mesh.

●●●

---

### Choose segments by nodes list

1. **n1(i), n2(i)...end** Node numbers of each segment to be chosen.

For each set of node numbers, the segment whose end points correspond to the given nodes will be chosen.

•••

## Choose segments line

1. **x1, y1, z1** *xyz*-coordinates [L] of the first end point of the line.
2. **x2, y2, z2** *xyz*-coordinates [L] of the second end point of the line.

Segments that fall on or close to the line are chosen. The command finds the two nodes closest to the end points of the line and then finds the group of connected line segments that form the shortest path between the two nodes.

•••

## Choose segments polyline

1. **npts** Number of points defining the polyline.
2. **x(i), y(i), z(i), i=1,npts** List of polyline point *xyz*-coordinates [L], which should be entered in order from one end of the polyline to the other.

Segments that fall on or close to the polyline are chosen. The command proceeds along the polyline, considering two points at a time. For each set of points it finds the two nearest nodes and then finds the group of connected line segments that form the shortest path between the two nodes.

•••

## Choose segments polyline by sheet

1. **nsheet** Node sheet number.
2. **npts** Number of points defining the polyline.
3. **x(i), y(i), i=1,npts** List of polyline point *xy*-coordinates [L], which should be entered in order from one end of the polyline to the other.

Segments that fall on or close to the polyline are chosen. The command proceeds along the polyline, considering two points at a time. For each set of points it finds the two nearest nodes in the specified sheet and then finds the group of connected line segments that form the shortest path between the two nodes.

•••

## Choose segments am node list

1. **npts** Number of points defining the polyline.
2. **node(i), sheet(i), i=1,npts** List of polyline point node and sheet numbers, which should be entered in order from one end of the polyline to the other.

Segments that fall on or close to the polyline are chosen. This instruction is intended to help to build horizontal wells/drains.

•••

---

### Choose segments xy between sheets

1. **x, y** *xy*-coordinates [L] to define a vertical line.
2. **isheet1, isheet2** Bottom and top sheet numbers (inclusive) to define the line.

The *xy*-coordinates together with the sheet numbers define the end points of a vertical line. Segments that fall on or close to the line are chosen. This instruction is intended to help to build vertical wells/drains.

•••

---

### Write chosen segments

1. **filename** Name of the file to which the chosen segment information will be written.

For each chosen segment, writes start and end node ids to the specified output file, one segment per line.

•••

## 2.4.3 Selecting Faces

---

### Allow internal faces

Causes **grok** to define internal faces, which cut through elements. By default, only the external faces (six orthogonal faces for 8-node blocks and five faces for 6-node prisms) are defined for the mesh.

•••

The following instructions may be used to alter the set of chosen faces.

---

### Clear chosen faces

All faces in the domain are flagged as *not* chosen. This command is recommended if you are unsure of which faces are chosen due to previously issued commands.

•••

## Clear chosen faces am

1. **filename** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Faces flagged as true in the file, that are between the top and bottom sheets (inclusive), and that are not oriented perpendicular to the sheets are flagged as *not* chosen.

•••

## Clear chosen faces top am

1. **filename** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description*.

Faces flagged as true in the file, that are in the top sheet, and are not oriented perpendicular to the top sheet are flagged as *not* chosen.

•••

## Clear chosen faces top by shp

1. **filename** Name of the shapefile without the file extension.

Faces in the top sheet whose centroids lie within a polygon defined by the shapefile are flagged as *not* chosen. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.*

•••

## Choose faces all

All faces in the domain are flagged as chosen. This command is useful if you wish to assign a property to all faces in the grid.

•••

## Choose faces x plane

1. **x1** $x$-coordinate [L] of the plane.
2. **ptol** Distance [L] from the plane.

Faces within distance **ptol** of the plane defined by the equation $x =$ **x1** will be chosen. This command is particularly useful when assigning boundary conditions to a specific face of a rectangular domain.

● ● ●

---

## Choose faces y plane

As above but for the $y$-plane.

● ● ●

---

## Choose faces z plane

As above but for the $z$-plane.

● ● ●

---

## Choose faces 3pt disk

`Scope: .grok`

1. **x1, y1, z1** $xyz$-coordinates [L] of a point on the disk.
2. **x2, y2, z2** $xyz$-coordinates [L] of a point on the disk.
3. **x3, y3, z3** $xyz$-coordinates [L] of disk center.
4. **radius** Radius [L] of the disk ($> 0$).
5. **tol** Distance [L] above and below the disk ($> 0$).

Faces whose centroids are within the cylindrical region bisected by the disk with height **tol** above and below the disk are chosen. Note that all three points on the disk must be distinct.

● ● ●

---

## Choose faces 3pt plane

1. **x1, y1, z1** $xyz$-coordinates [L] of the first point.
2. **x2, y2, z2** $xyz$-coordinates [L] of the second point.
3. **x3, y3, z3** $xyz$-coordinates [L] of the third point.

4. **ptol** Distance [L] from the plane.

Faces within a distance **ptol** of the plane defined by the three points will be chosen. This command allows you to choose planes of faces with an arbitrary orientation and is particularly useful for setting up a set of sloping fractures.

●●●

## Choose faces 3pt plane bounded

1. **x1, y1, z1** $xyz$-coordinates [L] of the first point.
2. **x2, y2, z2** $xyz$-coordinates [L] of the second point.
3. **x3, y3, z3** $xyz$-coordinates [L] of the third point.
4. **ptol** Distance [L] from the plane.
5. **x4, x5** $x$-range [L] of the block.
6. **y4, y5** $y$-range [L] of the block.
7. **z4, z5** $z$-range [L] of the block.

Faces within a distance **ptol** of the plane defined by the three points and within the rectangular block defined by the three ranges will be chosen.

●●●

## Choose faces block

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.

Faces whose centroids are within the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

●●●

## Choose faces block by layer

1. **x1, x2** $x$-range [L] of the block.

2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.
4. **nlaybot, nlaytop** Bottom and top element layer numbers.

Faces whose centroids are within the rectangular block which is defined by the three coordinate ranges, and which lie within the element layers defined by **nlaybot** and **nlaytop** are chosen. These layer numbers do not correspond to those given during grid generation but are simply defined by numbering each sheet of elements from 1 (bottom) to $n-1$ (top) where $n$ is the number of node sheets (2-D meshes) making up the grid.

This instruction is intended for grids that are regular in the $x$- and $y$-directions, but which have variable $z$-values for a given element layer. It can be used if the top and bottom elevations of a 3-D element layer vary spatially.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

●●●

## Choose faces sheet

1. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Faces that are between the two specified sheets (inclusive) and are not oriented perpendicular to the sheet will be chosen.

●●●

## Choose faces top

All faces in the top sheet of the domain will be chosen.

●●●

## Choose faces top block

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.

Faces in the top sheet whose vertices are within the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

•••

## Choose faces top block by centroid

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.

Faces in the top sheet whose centroid is within the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

•••

## Choose faces top from raster

1. **filename, (band)** Name of raster file and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).
2. **n1, n2** Range of raster values.

Faces in the top sheet whose centroids lie within the given raster class range are chosen. Selects all raster values that lie between **n1** and **n2** (inclusive).

•••

## Choose faces top shp

1. **filename** Name of the shapefile without the file extension.

Faces in the top sheet whose centroids lie within a polygon defined by the shapefile are chosen. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.*

•••

## Choose faces bottom

All faces in the bottom sheet of the domain will be chosen.

•••

## Choose faces front

Faces on the front of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Front faces are parallel to the $xz$-coordinate plane and have small $y$-coordinates.

●●●

## Choose faces back

Faces on the back of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Back faces are parallel to the $xz$-coordinate plane and have large $y$-coordinates.

●●●

## Choose faces left

Faces on the left side of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Left side faces are parallel to the $yz$-coordinate plane and have small $x$-coordinates.

●●●

## Choose faces right

Faces on the right side of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Right side faces are parallel to the $yz$-coordinate plane and have large $x$-coordinates.

●●●

## Choose faces top am

1. **filename** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description*.

Faces flagged as true in the file, that are in the top sheet, and are not oriented perpendicular to the top sheet are chosen.

●●●

## Choose faces top am common

1. **filename1** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description1*.
2. **filename2** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description2*.

Faces flagged as true in both files, that are in the top sheet, and are not oriented perpendicular to the top sheet are chosen.

•••

## Choose faces top am exclude

1. **filename1** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description1*.
2. **filename2** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description2*.

Faces flagged as true in the first file and flagged as false in the second file, that are in the top sheet, and are not oriented perpendicular to the top sheet are chosen.

•••

## Choose faces top for chosen elements

Faces are chosen if they are in the top sheet and the 3-D element they belong to is chosen.

•••

## Choose faces am

1. **filename** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Faces flagged as true in the file, that are between the top and bottom sheets (inclusive), and that are not oriented perpendicular to the sheets are chosen.

•••

## Choose faces vertical from am nodes

1. **filename** Name of the AlgoMesh chosen nodes file *am_prefix*`.nchos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

This instruction is intended for meshes that are generated from AlgoMesh 2-D meshes, and is used to choose faces that are oriented perpendicular to the mesh.

If a node is chosen in the 2-D mesh, then the nodes in the 3-D mesh that have the same *xy*-coordinates (i.e., that fall in the same column of nodes as the 2-D node) and between the top and bottom sheets (inclusive) will be chosen. A face is then chosen if all of its four nodes are chosen.

•••

## Choose faces vertical from polyline

1. **isheet1, isheet2** Bottom and top sheet numbers (inclusive), respectively.
2. **npts** Number of points defining the polyline.
3. **x(i), y(i), i=1,npts** *xy*-coordinates [L] of points on the polyline within a single sheet, which should be entered in order from one end of the polyline to the other.

Vertical faces that fall on or close to the polyline for the specified node sheet interval are chosen. The command proceeds along the polyline, considering two points at a time. For each set of points it finds the two nearest nodes and then finds the group of connected line segments that form the shortest path between the two nodes.

•••

## Choose faces vertical from polyline shp

1. **filename** Filename of polyline shapefile (`.shp`).
2. **isheet1, isheet2** Bottom and top sheet numbers (inclusive), respectively.

Vertical faces that fall on or close to the polyline for the specified node sheet interval are chosen. The command proceeds along the polyline, considering two points at a time. For each set of points it finds the two nearest nodes and then finds the group of connected line segments that form the shortest path between the two nodes.

•••

## Choose horizontal faces on layer

1. **nlayer** Element layer number.
2. **x1, x2** *x*-range [L] of the block.
3. **y1, y2** *y*-range [L] of the block.

Horizontal faces within the layer of elements numbered **nlayer** and within the rectangular block which is defined by the *x*- and *y*-range are chosen. This instruction can be used to select horizontal faces (e.g., to make fractures) when the elevation of a given layer of nodes is irregular.

•••

## Choose faces stairway

1. **x1, y1, z1** *xyz*-coordinates [L] of the first point.
2. **x2, y2, z2** *xyz*-coordinates [L] of the second point.
3. **x3, y3, z3** *xyz*-coordinates [L] of the third point.

Horizontal and vertical faces of an inclined plane that is defined by three points are chosen. This instruction is mainly designed for verification purpose of modelling results using inclined fractures. Note that if using this instruction, fracture velocities are multiplied by a correction factor that accounts for the longer path that contaminants have to travel from node to node.

•••

## Choose fracture faces block

1. **x1, x2** *x*-range [L] of the block.
2. **y1, y2** *y*-range [L] of the block.
3. **z1, z2** *z*-range [L] of the block.

Faces that are fracture elements whose centroids belong to the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

•••

## Choose face by nodes

1. **n1, n2, n3, n4** Node numbers of the face to be chosen.

The face whose vertices correspond to the given nodes will be chosen. Note that in order to select a triangular face simply set **n4** $= 0$.

•••

## Choose faces by nodes list

1. **n1(i), n2(i), n3(i), n4(i)...end** Node numbers for each face to be chosen.

For each set of node numbers, the face whose vertices correspond to the given nodes will be chosen. Note that in order to select a triangular face simply set **n4(i)** = 0.

• • •

## Clear chosen faces by nodes

As above except the face will be cleared (i.e., not chosen).

• • •

## Choose faces horizontal circle

1. **x_mid, y_mid, z_mid** *xy*-coordinates [L] of the centre of the circle and elevation of the circle.
2. **radius** Radius [L] of the circle.
3. **ptol** Vertical tolerance [L].

Faces within a vertical distance **ptol** of elevation **z_mid**, and within the circle with centre **x_mid, y_mid** and radius **radius** are chosen. This command allows you to choose faces in a domain that has a circular ground-plan.

• • •

## Write chosen faces

1. **filename** Name of the file to which the chosen face information will be written.

For each chosen face, writes the face number to the specified output file. Setting up complex fracture networks with combinations of Choose face commands can be very time consuming in **grok** and this step does not need to be repeated as long as the grid structure remains the same. This command is intended to be used in conjunction with the command Read chosen faces.

• • •

## Write chosen faces and host element numbers

1. **filename** Name of the file to which the chosen face and host element information will be written.

For each chosen face, writes the face number and associated 3-D element numbers to the specified output file. If the second element number is zero, then the face is on the outside of the 3-D domain.

•••

---

### Read chosen faces

1. **filename** Name of the file from which the chosen face information will be read.

Reads face numbers from the specified input file and sets each face as chosen. If you want only those faces read from the file to be chosen, then make sure to issue the command Clear chosen faces before using this command. Otherwise, the results will be merged with the current set of chosen faces. This command may be useful if you want to apply a certain set of fracture material properties to more than one group of faces at a time.

•••

---

### Echo chosen faces

Causes the current set of chosen face numbers to be written to the *prefix*`o.eco` file.

•••

## 2.4.4 Selecting Elements

The following instructions may be used to alter the set of chosen elements.

---

### Clear chosen elements

All elements in the domain will be flagged as *not* chosen. This command is recommended if you are unsure of which elements are chosen due to previously issued commands.

•••

---

### Clear chosen elements am

1. **filename** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Elements flagged as true in the file that are between the top and bottom sheets (inclusive) are flagged as *not* chosen.

•••

---

### Clear chosen elements by shp

1. **filename** Name of the shapefile without the file extension.
2. **nlay_bot, nlay_top** Bottom and top layer numbers.

Elements whose centroids lie within a polygon defined by the shapefile and that are between the top and bottom layers (inclusive) are flagged as *not* chosen. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.*

●●●

## Invert chosen elements

Chosen elements will be flagged as *not* chosen and the originally unchosen elements will be chosen.

●●●

## Choose elements all

All elements in the domain will be chosen. This command is useful if you wish to assign a property to all elements in the grid.

●●●

## Choose elements by zone

1. **zone** Zone number.

Elements that belong to the specified zone are chosen.

●●●

## Choose elements x plane

1. **x1** $x$-coordinate [L] of the plane.
2. **ptol** Distance [L] from the plane.

Elements within a distance **ptol** of the plane defined by the equation $x = \mathbf{x1}$ are chosen.

●●●

## Choose elements y plane

As above but for the $y$-plane.

•••

## Choose elements z plane

As above but for the $z$-plane.

•••

## Choose elements 3pt plane

1. **x1, y1, z1** $xyz$-coordinates [L] of the first point.
2. **x2, y2, z2** $xyz$-coordinates [L] of the second point.
3. **x3, y3, z3** $xyz$-coordinates [L] of the third point.
4. **ptol** Distance [L] from the plane.

Elements whose centroids are within a distance **ptol** of the plane defined by the three points are chosen. This command allows you to choose planes of elements with an arbitrary orientation.

•••

## Choose elements block

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.

Elements whose centroids are within the rectangular block defined by the three ranges are chosen. Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

•••

## Choose elements block by layer

1. **x1, x2** $x$-range [L] of the block.
2. **y1, y2** $y$-range [L] of the block.
3. **z1, z2** $z$-range [L] of the block.
4. **nlay_bot, nlay_top** Bottom and top element layer numbers.

Elements whose centroids are within the rectangular block which is defined by the three coordinate ranges, and which lie between the top and bottome layers (inclusive) are chosen. These layer numbers do not correspond to those given during grid generation but are simply defined by numbering each sheet of elements from 1 (bottom) to $n-1$ (top) where $n$ is the number of node sheets (2-D meshes) making up the grid.

This instruction is intended for grids that are regular in the $x$- and $y$-directions, but which have variable $z$-values for a given element layer. It can be used if the top and bottom elevations of a 3-D element layer vary spatially.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line, or point, respectively.

• • •

## Choose elements by layer

1. **nlay_bot, nlay_top** Bottom and top element layer numbers.

Elements that are between the bottom and top layers (inclusive) are chosen.

• • •

## Choose elements top

All elements that are in the top layer are chosen.

• • •

## Choose elements list

1. **filename** Name of the file that contains the list of element numbers.

Elements listed in the file are chosen. The input file consists of a list of element numbers, one entry per line. The file is read until end-of-file is reached.

• • •

## Choose elements xyz list

1. **filename** Name of the file that contains the list of $xyz$-coordinates [L].

Elements that contain a point in the file are chosen. The input file consists of a list of $xyz$-coordinates, one entry per line. The file is read until end-of-file is reached.

• • •

## Choose elements am

1. **filename** Name of the AlgoMesh chosen elements file *am_prefix*`.echos.`*description*.
2. **nsheet_bot, nsheet_top** Bottom and top sheet numbers.

Elements flagged as true in the file that are between the top and bottom sheets (inclusive) are chosen.

●●●

## Choose elements shp

1. **filename** Name of the shapefile without the file extension.
2. **nlay_bot, nlay_top** Bottom and top layer numbers.

Elements whose centroids lie within a polygon defined by the shapefile and that are between the top and bottom layers (inclusive) are chosen. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.*

●●●

The following instructions allow you to choose elements based on raster surfaces, as described in Appendix G. Since rasters are independent of the 3-D mesh, they do not have to be re-generated if the mesh changes, as is the case with GMS formatted surfaces that have values corresponding to each node in the 2-D mesh.

## Choose elements between raster surfaces

1. **rasterfile_bot** Name of the raster file for the bottom surface.
2. **rasterfile_top** Name of the raster file for the top surface.

The $xy$-coordinates of the element centroid are used to determine if it falls in a valid cell (i.e., a cell that has no missing values) in both raster files. If it does, the coordinates are used to interpolate a raster value from the cell's four corner values. If the $z$-coordinate of the element centroid is greater than the raster value of the bottom surface and less than the raster value of the top surface, then the element is chosen.

●●●

The following instructions use the same input data structure except that a single raster file is read and used for comparison:

Choose elements above raster surface
Choose elements below raster surface

The following instructions use the same input data structure except that there is an additional constraint that the element is chosen only if its current zone number is zero. They are intended to be used in conjunction with the instruction Assign zone zero:

Choose elements between raster surfaces, iprop zero
Choose elements above raster surface, iprop zero
Choose elements below raster surface, iprop zero

The following instructions use the same input data structure except that GMS scalar files are read instead of raster files. The element is chosen based on the $z$-coordinate of the first node in its node list:

Choose elements between gms surfaces
Choose elements above gms surface
Choose elements below gms surface

## Choose elements by thickness from raster

1. **filename** Name of the raster file that contains the thickness [L] data and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).

An element is chosen if the $z$-coordinate of its centroid is greater than the surface elevation minus the thickness at its 2-D centroid $(x_c, y_c)$.

●●●

## Choose elements horizontal circle

1. **x_mid, y_mid, z_mid** $xy$-coordinates [L] of the centre of the circle and elevation [L] of the circle.
2. **radius** Radius of the circle [L].
3. **ptol** Vertical tolerance [L].
4. **rtol** Horizontal tolerance [L].

Elements whose centroids are within a vertical distance **ptol** of elevation **z_mid**, and within a horizontal distance **rtol** of the circle with centre **x_mid, y_mid** and radius **radius** are chosen. This command allows you to choose elements in a domain that has a circular ground-plan.

●●●

## Write chosen elements

1. **filename** Name of file to which chosen element information will be written.

For each chosen element, writes the element number to the specified file, one entry per line.

●●●

---

### Write chosen elements 2d

1. **filename** Name of file to which chosen element information will be written.

For each chosen element, writes the 2-D element number to the specified file (without repeating any values), one entry per line.

●●●

---

### Write chosen elements xyz

1. **filename** Name of file to which chosen element information will be written.

For each chosen element, writes the *xyz*-coordinates [L] of the element centroid to the specified file, one entry per line.

●●●

## 2.5 Simulation Control Options

### 2.5.1 General

Once the grid generation step is completed, the pre-processor generates a set of data for a default problem by assuming *saturated, steady-state flow in a non-fractured, homogeneous porous medium.* The porous medium properties for the default problem are listed in Table 2.6. By default, the finite-element approach is used and a transport simulation is not done. If the default problem setup and material properties are acceptable, it is likely that the only additional data required to complete the definition of the problem are some flow boundary conditions, which can be assigned as described in Section 2.7.

---

### Transient flow

Causes **HydroGeoSphere** to perform a time-stepping, transient flow solution. Otherwise, a steady-state flow solution is computed.

●●●

## Unsaturated

Causes **HydroGeoSphere** to perform a variably-saturated flow solution. Otherwise, fully saturated flow conditions are assumed.

• • •

## Dual nodes for surface flow

Causes **HydroGeoSphere** to use the dual-node approach to define the surface flow domain. By default, the common node approach is used.

• • •

## Dual nodes for fracture flow

Causes **HydroGeoSphere** to use the dual-node approach to define the discrete fracture flow domain. By default, the common node approach is used.

• • •

## Dual nodes for channel flow

Causes **HydroGeoSphere** to use the dual-node approach to define the channel flow domain. By default, the common node approach is used.

• • •

## Dual nodes for wells

Causes **HydroGeoSphere** to use the dual-node approach to define the well domain. By default, the common node approach is used.

• • •

## Dual nodes for tiles

Causes **HydroGeoSphere** to use the dual-node approach to define the tile domain. By default, the common node approach is used.

• • •

## Defined flow

This command defines the flow conditions for the simulation based on the initial conditions and boundary conditions defined in the *prefix*`.grok` file and applies the predefined flow conditions for the entire solute transport simulation.

• • •

#### 2.5.1.1 Finite-Difference Options

---

### Finite difference mode

Causes **HydroGeoSphere** to use the finite difference approach instead of the default finite element method. See Section 3.3 in the Theory Manual for details.

• • •

The following command is recommended for transport simulations when the finite difference method is used.

---

### Compute fd cross terms

Causes **HydroGeoSphere** to compute cross terms explicitly when the finite difference method is used, which, by default, are ignored.

• • •

We also strongly recommend using the Control volume command when running transport simulations with either the finite element or the finite difference approach.

---

### Control volume

Causes the control volume finite element approach to be used instead of the default standard finite element approach. This option results in a conservative scheme with better numerical stability then the default approach. See Section 3.10 in the Theory Manual for details on the discretization of the solute transport equations.

• • •

#### 2.5.1.2 Matrix Solver

The matrix solution procedure consists of three phases: initialization, preconditioning, and solution. The instructions presented here can be used to control the preconditioning and solution phases in order to increase the efficiency of the model.

The default preconditioning scheme is level-based factorization without red/black system reduction. Improving model performance requires a good understanding of these options, so if you are unsure, you should just use the default settings. However, if you decide to experiment with the solver preconditioning parameters, here are a few suggestions for doing so:

1. For transient, simple (i.e., small number of nodes) problems, level-based preconditioning works better, because the static data structure analysis does not need to be done for each time step.
2. For steady state or complex problems, drop tolerance preconditioning works better, because the solver spends most of its time in the solution phase, not the preconditioning phase.
3. For a very smoothly varying solution (e.g., a weakly stressed, homogeneous property field) red/black reduction will speed convergence.

## Level of fill

1. **level** Level of fill.

Assigns the level of fill to be preserved in level-based factorization, which defaults to zero. If drop tolerance preconditioning is used, this value is not used.

●●●

## Red black reduction

Causes the solver to use red black reduction. This option can be used with either level-based or drop tolerance preconditioning.

●●●

## Drop tolerance preconditioning

Causes the solver to use drop tolerance preconditioning, which preserves elements in the factorization based on their size.

●●●

## Drop tolerance threshold

1. **thresh** Drop tolerance threshold [-].

Assigns a new drop tolerance threshold. The default threshold is 0.1.

●●●

Once the preconditioning phase is complete, the matrix equation can be solved using several acceleration techniques. The following command can be used to change the solver method:

## Solver acceleration technique

1. **iaccel** Type of acceleration to use.

Sets the acceleration technique used by the linear solver, which defaults to 3 (CGSTAB-P). Appropriate values are 0 (CG, for symmetric matrices only), 1 (OrthoMin), 2 (CGS), 3 (CGSTAB-P), or 4 (GMRES). If unsure, don't use this command.

● ● ●

## No matrix scaling

By default the rows of the matrix equation $Ax = b$ are scaled by the inverse of the diagonal elements of $A$ prior to computing the preconditioner, that is,

$$D^{-1}Ax = D^{-1}b, \quad D = \text{diag}(A)$$

As a result, the elements on the main diagonal of the scaled matrix are all equal to one. Matrix scaling, which is a form of preconditioning, is recommended since it typically improves performance of the solution phase. This command tells the solver not to use matrix scaling preconditioning.

● ● ●

When solving difficult problems it is possible that the solver acceleration methods discussed above may "stall". A solve is considered stalled if the residual does not change by a certain amount over a specifed number of iterations. Stall checking is available for solver acceleration methods that benefit from restarting if stalling is detected, in this case, CGS and CGSTAB-P. If an acceleration method stalls, then it is automatically restarted. Note that restarting a solve is indicated to the user only when running with elevated solver diagnostic messages via the command Flow solver detail with the detail level set to one. The stall range, number of iterations required to flag a stall, and the number of automatic restarts are controlled by the following commands.

## Range for stall

1. **rstall** Residual tolerance [-] to determine a stalled solve.

Sets the tolerance on the relative residual to determine if a solver has stalled. If $k$ is the current solver iteration number, $\ell$ is the number of iterations to flag a stall, and $r_j$ is the residual vector at the $j$th solver iteration, then the solver is deemed to have stalled if the following condition is met:

$$r_{k,\min} > 0 \quad \text{and} \quad \frac{|r_{k,\max} - r_{k,\min}|}{r_{k,\min}} < \textbf{rstall},$$

where

$$r_{k,\max} = \max_{i=0,\ldots,\ell-1} \|r_{k-i}\|_2 \quad \text{and} \quad r_{k,\min} = \min_{i=0,\ldots,\ell-1} \|r_{k-i}\|_2$$

By default this value is set to $10^{-3}$.

• • •

## Stall check parameter

1. **stall_check** Number of iterations required to flag a stall.

Sets the number of iterations required to flag a stall. Checking for a stalled solve will compare the current residual to the residual **stall_check** iterations ago. By default this value is set to 15. Note that stall detection and solver restarting can be disabled simply by setting this parameter to zero.

• • •

## Number of stall cycles

1. **nstall** Maximum number of solver restarts due to stalling.

Sets the maximum number of times that a solver can be restarted due to stalling. By default this value is set to 30. Note that a solve will not be restarted due to a stall once this number is exceeded.

• • •

### 2.5.2 Timestep Control

Before discussing the available instructions for controlling the behaviour of a transient solution, some background information is required. The pre-processor **grok** generates a list of target times that are derived from the following sources:

- Times specified by the user to meet timestep constraints.
- Times specified by the user to meet output requirements.
- Times at which transient boundary condition values change.

This target time list is passed to **HydroGeoSphere**, which uses it to produce timestep values. As discussed in Section 2.5.2.1, adaptive timestepping can be used to adjust the timestep values based on changes in head, saturation, and/or concentration as the solution progresses. Without adaptive timestepping, **HydroGeoSphere** proceeds from one target time to the next.

**Important:** If a Newton solve fails with adaptive timestepping enabled, then the current timestep is reduced and the Newton solve is redone. However, if a Newton solve fails without adaptive timestepping, then the simulation fails.

## Target times

1. **target_time(i)...end** Target times [T] list.

Listed times are added to the current set of target times.

●●●

## Generate target times

1. **tstart** Start time [T].
2. **delta** Initial time step size [T].
3. **tinc** Time step multiplier [-].
4. **dtmax** Maximum time step size allowed [T].
5. **tend** End time [T].

New target times are generated from the start time **tstart** to end time **tend** by repeatedly adding the time step **delta**, which is increased each time by the multiplier **tinc** until it reaches a maximum size of **dtmax**.

●●●

## Output times

1. **output_time(i)...end** Output time [T] list.

Listed times are added to the current set of output times (i.e., times for which you want detailed output). Note that these values automatically become part of the target time list.

●●●

The following instructions can be used to modify the timestepping behaviour of a transient solution within the adaptive timestepping framework (they do not apply if adaptive timestepping is disabled):

## Initial time

1. **tinit** Initial time [T].

Assigns a new value for the initial time, which defaults to zero. This is useful if you are restarting the simulation and want to index the times used to an earlier run.

●●●

## Initial time from binary output file

1. **filename** Name of binary output file from which the initial time [T] is read.

Assigns a new value for the initial time, which is read from the header of a previously generated binary ouput file. For flow-only simulations we recommend using the porous medium head output file *prefix*`o.head_pm.`*suffix*. Otherwise, for transport simulations we recommend using the porous medium concentration output file *prefix*`o.conc_pm.`*species.suffix*.

●●●

## Initial timestep

1. **dtinit** Initial timestep size [T].

Assigns a new value for the initial timestep, which defaults to 0.01 time units.

●●●

## Minimum timestep

1. **dtmin** Minimum timestep size [T].

Assigns a new value for the minimum timestep size, which defaults to $10^{-10}$ time units. If the timestep becomes smaller than this value as a result of the adaptive timestepping procedure, **HydroGeoSphere** will stop and issue a diagnostic message.

●●●

## Maximum timestep

1. **dtmax** Maximum timestep size [T].

Assigns a new value for the maximum timestep size, which defaults to $10^{25}$ time units.

●●●

## Time varying maximum timestep

1. **time(i), max_timestep(i)...end** Simulation time [T] and maximum timestep size [T] list.

This command assigns a time varying value for the maximum timestep size. For each simulation time $t$, the maximum timestep size, $\Delta t_{\max}$, is defined as

$$\Delta t_{\max} = \begin{cases} \Delta t_{\max,i}, & t \in [t_i, t_{i+1}),\ 1 \le i \le n-1 \\ \Delta t_{\max,n}, & t \ge t_n \end{cases}$$

where $n$ is the size of the input list. Note that $\Delta t_{\max}$ is unchanged for all $t < t_1$.

• • •

#### 2.5.2.1 Adaptive Timesteps

If required, **HydroGeoSphere** can modify timestep values as the solution proceeds, based on the transient behaviour of the system (see Equation 3.101 and Appendix C). The following instructions can be used to activate this feature and to set targets for specific variables (e.g., pressure head, saturation, etc.). These targets are used to modify timestep size as the solution proceeds.

---

### Head control

1. **dhead_allowed** Maximum desired absolute change in nodal head [L] during any time step.

• • •

---

### Water depth control

1. **ddepth_allowed** Maximum desired absolute change in nodal surface water depth [L] during any time step.

• • •

---

### Saturation control

1. **dsat_allowed** Maximum desired absolute change in nodal saturation [-] during any time step.

• • •

---

### Newton iteration control

1. **nnri_allowed** Maximum desired number of Newton–Raphson iterations during any time step.

●●●

## DDF Picard iteration control

1. **npicard_allowed** Maximum desired number of density-flow Picard iterations during any time step.

●●●

## Concentration control

1. **dconc_allowed** Maximum desired absolute change in nodal concentration [M L$^{-3}$] during any time step.

●●●

## Concentration control, multi-species

1. **dconc_allowed(i), i=1,nspeciesmob** Maximum desired absolute change in nodal concentration [M L$^{-3}$] for the $i$th species during any time step.

●●●

## Mass change control

1. **dmass_change_allowed** Maximum desired absolute change in mass [M] during any time step.

●●●

## Mass error control

1. **dmass_error_allowed** Maximum desired absolute mass error [M] during any time step.

●●●

The following instructions are used to generate a timestep multiplier according to Equation 3.101; see also Appendix C. The multiplier is constrained to lie between a lower and upper bound (inclusive), which by default is the interval $[0.5, 2]$. If you would like to modify these limits, you may do so via the following instructions:

---

#### Minimum timestep multiplier

1. **dtmul_min** Minimum timestep multiplier [-].

Assigns a new value for the minimum timestep multiplier, which defaults to 0.5.

•••

---

#### Maximum timestep multiplier

1. **dtmul_max** Maximum timestep multiplier [-].

Assigns a new value for the maximum timestep multiplier, which defaults to 2.

•••

### 2.5.3 Saturated Flow

---

#### Pressure head input

Causes all heads which are input to be treated as pressure heads instead of hydraulic heads.

•••

---

#### Freshwater pressure head

Causes all heads which are input and output to be treated as freshwater pressure heads instead of hydraulic heads.

•••

---

#### Fluid pressure

Causes all heads which are input and output to be treated as fluid pressures instead of hydraulic heads.

•••

## Flow time weighting

1. **tw** Time-weighting factor [-] for the flow solution.

Assigns a new value for the time-weighting factor for the flow solution, which defaults to 1.0. Values must be greater than 0.0 and less than or equal to 1.0.

●●●

## Flow solver maximum iterations

1. **maxfit** Maximum number of flow solver iterations.

Assigns a new value for the maximum number of flow solver iterations, which defaults to 2000.

●●●

The flow solver solves a linear system of the form $Ax = b$, where $A$ is the coefficient matrix, $x$ is the vector of unknowns, and $b$ is a vector of known values. The linear system is solved approximately, with convergence based on the infinity-norm of the residual vector $b - Ax$. If abstol denotes the absolute convergence tolerance and reltol denotes the relative convergence tolerance, then the flow solver convergence criterion is given by

$$\|b - Ax\|_\infty \leq \max(\text{abstol}, \text{reltol} \cdot \|b\|_\infty).$$

When simulating variably saturated flow, the norm of the right-hand side vector $b$ is equivalent to the norm of the Newton residual. Hence, the flow solver convergence tolerance scales linearly with the size of the Newton residual when reltol $> 0$. The absolute and relative convergence tolerances can be specified by the following commands.

## Flow solver convergence criteria

1. **abstol** Flow solver absolute convergence tolerance [$\text{L}^3$ $\text{T}^{-1}$].

Assigns a new value for the flow solver absolute convergence tolerance, which defaults to $10^{-10}$.

●●●

## Flow solver relative convergence criteria

1. **reltol** Flow solver relative convergence tolerance [-].

Assigns a new value for the flow solver relative convergence tolerance, which defaults to zero. When specifying **reltol**, we recommend a value in the range of $10^{-4}$ to $10^{-6}$ as a good starting point.

●●●

---

## Flow solver detail

1. **isolv_info** Flow solver detail level.

Assigns a new value for the flow solver detail level, which defaults to 0. This value controls the level of detail of solver performance information printed to the screen and listing file, and can have values of 0 (no information) or 1 (full information).

●●●

### 2.5.4 Variably-Saturated Flow

By default, **HydroGeoSphere** uses the upstream weighting scheme (weighted harmonic mean) for relative permeability, with an upstream weighting factor value of 1.0. The default behaviour can be changed with the following commands:

---

## Upstream weighting factor

1. **upwfactor** Upstream weighting factor [-].

Assigns a new value for the upstream weighting factor, which defaults to 1.0. This value should be in the range of 0.5 (central weighting) to 1.0 (upstream weighting).

●●●

---

## Central weighting

Causes **HydroGeoSphere** to use central weighting (weighted arithmetic mean) instead of upstream weighting.

●●●

We recommend that the following command always be used for models that define a surface flow domain on a triangular mesh.

---

## Integrated finite difference for overland flow

Causes **HydroGeoSphere** to use an incenter approach for computing conductance values in the control volume finite element method. If a conductance in a right triangle element is computed using a circumcenter approach (the default), then the conductance value along one edge of the triangle can be zero, which blocks the flow of water and generates a water reservoir. By using this command, the water accumulation problem for meshes that contain right triangles can be avoided.

•••

#### 2.5.4.1 Newton Iteration Parameters

The following parameters can be used to control the Newton–Raphson iteration scheme for solution of the variably-saturated flow problem as described in Section 3.14.2 of the Theory Manual.

---

## Jacobian epsilon

1. **epsilon** Jacobian epsilon [L].

Assigns a new value for the Jacobian epsilon, which defaults to $10^{-4}$. The Jacobian epsilon is the shift in pressure head used to numerically compute the derivatives in the Jacobian matrix. As a rule of thumb, a value equal to $10^{-5}$ times the average pressure head in the domain is recommended.

•••

---

## Newton maximum iterations

1. **maxnewt** Maximum number of Newton iterations.

Assigns a new value for the maximum number of Newton iterations, which defaults to 15. If this number is exceeded during a timestep, then the current timestep length is reduced by half and a new solution is attempted.

•••

---

## Newton minimum iterations

1. **minnewt** Minimum number of Newton iterations.

Assigns a new value for the minimum number of Newton iterations, which defaults to 0. Convergence of the Newton iteration can be achieved only after it has performed the minimum number of iterations.

•••

## Newton absolute convergence criteria

1. **delnewt** Newton absolute convergence tolerance [L].

Assigns a new value for the Newton absolute convergence tolerance, which defaults to $10^{-5}$. Convergence of the Newton iteration is achieved when the maximum absolute nodal change in pressure head for one Newton iteration is less than this value.

• • •

## Newton residual convergence criteria

1. **resnewt** Newton residual convergence tolerance [$L^3$ $T^{-1}$].

Assigns a new value for the Newton residual convergence tolerance, which defaults to $10^{-8}$. Convergence of the Newton iteration is achieved when the maximum absolute nodal residual for one Newton iteration is less than this value.

• • •

The following command can be used to define an additional convergence criterion based on the 2-norm of the Newton residual, with convergence of the Newton iteration declared when any of these criteria are satisfied.

## Newton residual 2-norm convergence criteria

1. **resnewt** Newton residual convergence tolerance [$L^3$ $T^{-1}$] in the 2-norm.

Assigns a new value for the Newton residual convergence tolerance in the 2-norm, which defaults to zero (test is inactive). Convergence of the Newton iteration is achieved when the 2-norm of the Newton residual for one Newton iteration is less than this value.

• • •

## Minimum relaxation factor for convergence

1. **minrelfac_convergence** Minimum relaxation factor [-] to declare convergence.

Assigns a new minimum relaxation factor to declare convergence of the Newton iteration, which defaults to 0.95. Convergence of the Newton iteration can only be achieved when the computed relaxation factor is larger than this minimum value.

• • •

## Newton maximum update for head

1. **NR_dhtol** Newton maximum update for head [L].

Assigns a new value for the Newton maximum update for head, which defaults to 1.0. This value is used to calculate the underrelaxation factor $\omega_r$ according to

$$\omega_r = \frac{\mathbf{NR\_dhtol}}{\max(dh_r)}$$

$$h_r = h_{r-1} + \omega_r \cdot dh_r$$

where $\max(dh_r)$ is the computed maximum update for head in $r$th Newton iteration and $h_r$ is the head flow solution after $r$ iterations. As **NR_dhtol** becomes smaller, the Newton solution becomes more stable but with possibly more iterations. For highly nonlinear problems for which Newton linearization easily fails to converge, it is recommended to set this value smaller.

●●●

## Newton maximum update for depth

1. **NR_ddtol** Newton maximum update for depth [L].

Assigns a new value for the Newton maximum update for water depth, which defaults to $10^{-2}$. The update equations are the same as those for command Newton maximum update for head, but are applied to water depth.

●●●

## Newton absolute maximum residual

1. **NR_max_resnorm** Newton absolute maximum residual value [$L^3$ $T^{-1}$].

If after any iteration the 2-norm of the Newton residual exceeds **NR_max_resnorm**, then the Newton iteration is restarted with a smaller timestep. By default this tolerance is set to zero, which is treated the same as $\infty$.

●●●

## Newton maximum residual increase

1. **NR_resnorm_fac** Newton maximum residual increase [-].

Assigns a new value for the Newton maximum residual increase, which defaults to $10^{30}$. If after any iteration the 2-norm of the Newton residual increases by a factor of more than **NR_resnorm_fac**, then the Newton iteration is restarted with a smaller timestep.

•••

## Remove negative coefficients

Forces negative inter-nodal conductances to zero. Negative inter-nodal conductances result in inter-nodal flow from lower to higher heads and can cause oscillatory behavior during Newton iterations (Letniowski and Forsyth, 1991).

•••

## No nodal flow check

Turns off the nodal flow check, which is on by default. The nodal flow check ensures that the transport simulation, which uses the flow solution for advective transport, will not result in solutions that contain spurious local maxima and minima (McLaren et al., 2000). In cases where only a flow solution is being computed (i.e., no transport), the nodal flow check is unnecessary and should be turned off.

•••

## Nodal flow check tolerance

1. **n_flow_check_tol** Nodal flow check tolerance [-].

Assigns a new nodal flow check tolerance, which defaults to $10^{-2}$. The tolerance is used to constrain a particular measure of nodal flow error, $\text{flow}_i^{\text{error}}$, such that

$$\max_i \text{flow}_i^{\text{error}} \leq \textbf{n\_flow\_check\_tol}$$

where the index $i$ ranges over all nodes in the porous media and dual continuum domains. The definition of the error term and its derivation are given by McLaren et al. (2000).

•••

## Underrelaxation factor

1. **under_rel** Underrelaxation factor [-].

Assigns a new value for the underrelaxation factor for the Newton iteration, which defaults to 1. This value can range from 0 (full underrelaxation) to 1 (no underrelaxation).

•••

## Compute underrelaxation factor

Causes the underrelaxation factor $\omega$ [-] to be computed according to the following method described by Cooley (1983):

$$\omega_{r+1} = \begin{cases} \frac{3+s}{3+|s|} & \text{if } s \geq -1 \\ \frac{1}{2|s|} & \text{if } s < -1 \end{cases}$$

where

$$s = \begin{cases} \frac{e_{r+1}}{e_r \omega_r} & \text{if } r > 1 \\ 1 & \text{if } r = 1 \end{cases}$$

In the equations above, $r$ and $r+1$ represent the previous and current iteration level, $\omega_r$ and $\omega_{r+1}$ represent the underrelaxation factor for the previous and current iteration levels, and $e$ represents the maximum value of the largest difference between head values for two successive iterations, $e_r = \max_i |\psi_i^r - \psi_i^{r-1}|$.

• • •

## Compute underrelaxation factor limit

1. **dellim** Maximum computed underrelaxation factor [-].

Assigns a new upper limit on the computed underrelaxation factor, which defaults to 1000. A value of 10 times the system domain thickness is recommended.

• • •

## Minimum relaxation factor allowed

1. **min_relfac_allowed** Minimum computed underrelaxation factor [-].

Assigns a new lower limit on the computed underrelaxation factor, which defaults to $10^{-3}$. If not at the first timestep and the computed underrelaxation factor is less than this value, then the current timestep is cut in half and the Newton–Raphson iteration loop is restarted.

• • •

## Newton information

Causes **HydroGeoSphere** to write more detailed information to the listing file about the performance of the Newton iteration process.

• • •

## Detailed runtime information

Causes **HydroGeoSphere** to generate two Tecplot ASCII output files named *prefix*`o.newton_runtime_info.dat` and *prefix*`o.adaptive_timestep_info.dat` that contain detailed information about the flow solution Newton iterations and adaptive timestepping, respectively, at each timestep. The variables in these files are discussed in detail in Appendix C, which covers the **HydroGeoSphere** run-time timestep output.

●●●

### 2.5.5 Transport

#### Transport time weighting

1. **twc** Time-weighting factor [-] for the transport solution.

Assigns a new value for the time-weighting factor for the transport solution, which defaults to 1.0. Values can range from 0.0 (explicit) to 0.5 (central or Crank–Nicholson) or 1.0 (fully implicit). Fully implicit time-weighting is less prone to exhibit oscillations but more prone to numerical smearing than central time-weighting.

●●●

#### Transport solver maximum iterations

1. **maxtit** Maximum number of transport solver iterations.

Assigns a new value for the maximum number of transport solver iterations, which defaults to 2000.

●●●

The transport solver solves a linear system of the form $Ax = b$, where $A$ is the coefficient matrix, $x$ is the vector of unknowns, and $b$ is a vector of known values. The linear system is solved approximately, with convergence based on the infinity-norm of the residual vector $b - Ax$. If abstolc denotes the absolute convergence tolerance and reltolc denotes the relative convergence tolerance, then the transport solver convergence criterion is given by

$$\|b - Ax\|_\infty \leq \max(\text{abstolc}, \text{reltolc} \cdot \|b\|_\infty).$$

The absolute and relative convergence tolerances can be specified by the following commands.

#### Transport solver convergence criteria

1. **abstolc** Transport solver absolute convergence tolerance [M $L^{-3}$ $T^{-1}$].

Assigns a new value for the transport solver absolute convergence tolerance, which defaults to $10^{-10}$.

●●●

## Transport solver relative convergence criteria

1. **reltolc** Transport solver relative convergence tolerance [-].

Assigns a new value for the transport solver relative convergence tolerance, which defaults to zero. When specifying **reltolc**, we recommend a value in the range of $10^{-4}$ to $10^{-6}$ as a good starting point.

●●●

## Transport solver detail

1. **isolv_infoc** Transport solver detail level.

Assigns a new value for the transport solver detail level, which defaults to 0. This value controls the level of detail of solver performance information printed to the screen and listing file, and can have values of 0 (no information) or 1 (full information).

●●●

## Upstream weighting of velocities

1. **almax, btmax, gammax** Upstream weighting factors [-] in the $x$-, $y$-, and $z$-directions, respectively.

Causes upstream weighting of velocities, which by default is disabled. Values can range from 0 (no upstream weighting) to 1 (full upstream weighting). Note that these variables do not apply to the control volume case, where full upstream weighting is always used.

●●●

## Advective solute exchange only

This instruction affects transport in either coupled surface/subsurface or fracture/subsurface systems and causes **HydroGeoSphere** to neglect dispersive/diffusive exchange between the two domains. It is intended to be used to gauge the relative importance of advective versus dispersive exchange or when comparing **HydroGeoSphere** to codes that neglect dispersive exchange.

●●●

## Peclet number

1. **pectol** Peclet number [-].

Assigns a new value for the Peclet number, which defaults to $10^{20}$. The Peclet number does not influence the solution in any way, but is merely used to generate warning messages. A check against the tolerance **pectol** is performed for all elements and species at each timestep, and exceedances are noted in the *prefix*`o.lst` file. You can change the level of output to the *prefix*`o.lst` file via the command Peclet verbose. The Peclet number in the $x$-direction is defined as

$$Pe = \frac{v_x \cdot \Delta x}{D_{xx}}$$

where $v_x$ is a flow velocity, $\Delta x$ is an element length, and $D_{xx}$ is a dispersion coefficient (the $y$- and $z$-directions are defined analogously). It is a measure of the adequacy of mesh fineness, with large numbers indicating poor spatial discretization. The large default value serves to suppress the generation of warning messages.

●●●

## Peclet verbose

Causes **HydroGeoSphere** to write Peclet number exceedance warning messages on a per species basis. By default, a single warning message is generated that gives the maximum computed Peclet number in each dimension for all species.

●●●

## Output peclet number

Causes **HydroGeoSphere** to write the Peclet number for each element to the file *prefix*`o.Peclet_pm.0001` and the diffusion Peclet number for each element to the file *prefix*`o.DiffPeclet_pm.0001`, at the first timestep only.

●●●

## Courant number

1. **courtol** Courant number [-].

Assigns a new value for the Courant number, which defaults to $10^{20}$. The Courant number does not influence the solution in any way, but is merely used to generate warning messages. A check against the tolerance **courtol** is performed for all elements and species at each timestep, and exceedances are noted in the *prefix*`o.lst` file. You can change the level of output to the *prefix*`o.lst` file via the command Courant verbose. The Courant number in the $x$-direction is defined as

$$C = \frac{v_x \cdot \Delta t}{\Delta x}$$

where $v_x$ is a flow velocity, $\Delta x$ is an element length, and $\Delta t$ is the timestep length (the $y$- and $z$-directions are defined analogously). It is a measure of the adequacy of timestep size, with large numbers indicating poor temporal discretization. The large default value serves to suppress the generation of warning messages.

●●●

## Courant verbose

Causes **HydroGeoSphere** to write Courant number exceedance warning messages on a per species basis. By default, a single warning message is generated that gives the maximum computed Courant number in each dimension for all species.

●●●

### 2.5.6 Density-Dependent Flow Solution

The following command can be used to control the Picard iteration for the solution of the weakly nonlinear density-dependent flow problem.

## Picard convergence criteria

1. **head_tol** Convergence tolerance [L] for the maximum absolute difference in head for the Picard iteration. Default value of $10^{-5}$.

2. **conc_tol** Convergence tolerance [M L$^{-3}$] for the maximum absolute difference in concentration for the Picard iteration. Default value of $10^{-5}$.

3. **maxits** Maximum number of iterations for the Picard iteration. If the maximum number of iterations is reached without satisfying the stopping criteria, then the timestep is cut in half and the iteration is started over. Default value of 100.

Assigns new values to the Picard convergence criteria. Note that both the head and concentration stopping criteria must be satisfied in order to stop the Picard iteration.

●●●

## Steady state density dependent flow

Causes **HydroGeoSphere** to perform a steady-state density dependent flow solution. The solution is computed via pseudo-transient continuation and parameter continuation methods. The numerical scheme uses the parameters: density, timestep size, and relaxation factor to constrain changes in the state variables (i.e., head and concentration) to achieve a solution. This command requires that the input files `denmarch.in` and `multiplication_factor.dat` be present in the simulation folder.

The file `denmarch.in` contains the input parameters required by the numerical scheme. An example file is provided below.

```
__MaxDensity_TimeStep_PiccardIterWeighting
0.025 ! Maximum density (gamma = rho_max/rho_0 - 1)
1e4   ! Minimum timestep size for convergence
1.0   ! Minimum relaxation factor for Picard iteration convergence

__Tolerances:_HeadChange_ConcentrationChange
1e-6  ! Convergence tolerance for head change
1e-6  ! Convergence tolerance for concentration change

__Initial_MultiplicationFactors:_m1_m2_m3
1.0   ! Multiplication factor 1 (m1) for density increase
1.0   ! Multiplication factor 2 (m2) for timestep increase
1.0   ! Multiplication factor 3 (m3) for relaxation factor

__PenaltyFactors:_TimeStep_Relaxation
1.0   ! Penalty factor for timestep size
1.0   ! Penalty factor for relaxation factor

__Initial_DensityIncrease_MaxDensityIncrease
1e-3  ! Initial density increase (gamma)
1e-3  ! Maximum density increase (gamma)
```

We note that a steady-state density dependent flow solution is accepted when the maximum absolute change in both the head and concentration are less than their respective tolerances. In practice, we recommend using convergence tolerances no smaller than $10^{-10}$.

The file `multiplication_factor.dat` contains three tables for determining the multiplication factors: $m_1$, $m_2$, and $m_3$. In this file, the first table is for $m_1$, the second table is for $m_2$, and the third table is for $m_3$. Each table consists of two columns, the first being the number of Picard iterations and the second being a multiplication factor. The multiplication factors are computed from their respective tables via linear interpolation. An example file is provided below.

```
 6       ! Number of points for m1 table
 0  1.5
10  1.2
20  1.0
30  0.5
40  0.25
```

```
50  0.1
 0.25   ! Penalty for timestep size
 6      ! Number of points for m2 table
 0  1.5
10  1.2
20  1.0
30  0.5
40  0.25
50  0.1
 0.25   ! Penalty for relaxation factor
 6      ! Number of points for m3 table
 0  1.5
10  1.2
20  1.0
30  0.5
40  0.25
50  0.1
```

•••

The following commands apply globally when running density-dependent flow simulations.

## Constant viscosity

Forces the fluid viscosity to be held constant. The value of the fluid viscosity can be set by the command Reference fluid viscosity.

•••

## Oberbeck boussinesq assumption

1. **level** Level of Oberbeck–Boussinesq approximation, either 1 or 2.

Sets the level of the Oberbeck–Boussinesq approximation, which by default is set to 1. The levels are defined as follows:

1 Density variations are NOT considered for either the fluid or solute mass balance.

2 Density variations are considered for the fluid mass balance but NOT the solute mass balance.

Note that this command applies only to unsaturated flow in porous media.

•••

### 2.5.7 Heat Transfer

To enable heat transfer, you must define a temperature species in the solute definition block via the command Temperature species. The thermal properties of the solids are specified in the material property file. The following instructions may be used to define the heat transfer solution.

---

#### Thermal conductivity of air

1. **k_air** Thermal conductivity [W m$^{-1}$ K$^{-1}$] of the air phase.

Assigns a uniform value to the thermal conductivity of air.

●●●

---

#### Specific heat capacity of air

1. **c_air** Specific heat capacity [J kg$^{-1}$ K$^{-1}$] of the air phase.

Assigns a uniform value to the specific heat capacity of air.

●●●

---

#### Density of air

1. **rho_air** Density [kg m$^{-3}$] of the air phase.

Assigns a uniform value to the density of air.

●●●

---

#### Thermal conductivity of water

1. **k_l** Thermal conductivity [W m$^{-1}$ K$^{-1}$] of the liquid phase.

Assigns a uniform value to the thermal conductivity of water. If this instructions is used, the thermal conductivity of water is assumed constant and equal to $k_l$. It is therefore not calculated from the water temperature, which is the default setting.

●●●

---

#### Specific heat capacity of water

1. **c_l** Specific heat capacity [J kg$^{-1}$ K$^{-1}$] of the liquid phase.

Assigns a uniform value to the specific heat capacity of water. If this instructions is used, the specific heat capacity of water is assumed constant and equal to $c_l$. It is therefore not calculated from the water temperature, which is the default setting.

● ● ●

## Mechanical heat dispersion

Causes mechanical heat dispersion to be simulated. The default is no mechanical heat dispersion.

● ● ●

The following instruction can be used to define initial temperature profile.

## Initial temperature profile

1. **temp_top** Temperature [°C] at the top of the domain.
2. **temp_grad** Prevailing geothermal gradient [L$^{-1}$ K].

Calculates a depth profile of temperature and assigns the values to the initial temperature.

● ● ●

The following instructions can be used to define a heat source boundary condition.

## Zero order source

1. **npanel** Number of panels in the time-variable, zero-order source function.
2. **ton(i), toff(i), prate(i,j), j=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], heat production rate [M L$^{-1}$ T$^{-3}$].

Nodes that belong to the chosen zones are assigned zero-order source boundary conditions.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term specified for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to one, **ton** to zero, and **toff** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **prate** should be included. For example, with three panels and two species the input format is

```
3 ! npanel
ton1 toff1 prate11 prate12
ton2 toff2 prate21 prate22
```

```
ton3 toff3 prate31 prate32
```

• • •

---

### Exponential zero order source

1. **heat_q_zero** Heat production [M $L^{-1}$ $T^{-3}$] at time $t = 0$.
2. **heat_constant** Exponential function growth constant [$T^{-1}$].

Nodes in the chosen zones area assigned an exponentially decreasing zero-order heat source boundary conditions.

• • •

### 2.5.8 Inactive Elements

These instructions can be used to discretize irregular boundaries with block elements by deactivating portions of the grid, where all elements become inactive for both the flow and transport simulation. Elemental assembly is skipped and all nodes that only belong to the inactive element, and are not at all connected to active elements, are assigned default values of head and concentration equal to $-9999$. This option is similar to what is done in MODFLOW to specify inactive cells.

---

### Make element inactive

All chosen elements will become inactive.

• • •

---

### Make zone inactive

All elements in the current set of chosen zones will become inactive.

• • •

For a 2-D slice made of 4-node rectangular elements, the following instruction can be used to make elements inactive:

---

### Make element inactive using shapefile

1. **filename** Name of the shapefile without the file extension.

2. **unproject_file** Logical value (T/F), which if true, causes **grok** to read grid unprojection data and apply it to the data read from the shapefile. Note: this option is no longer supported an should always be set to false (F).

3. **project_file** Logical value (T/F), which if true, causes **grok** to read grid projection data and apply it to the data read from the shapefile. Note: this option is no longer supported an should always be set to false (F).

4. **outside** Logical value (T/F), which if true, causes elements located outside the area defined in the shapefile to become inactive. Otherwise, elements located inside the area will become inactive.

•••

## Write inactive elements to file

1. **inactive_file** Name of the file to which inactive element information is written.

Writes element numbers of inactive elements to the specified output file, one entry per line. Setting up inactive elements with the Make element inactive using shapefile instruction can be time consuming in **grok** and this step does not need to be repeated as long as the grid structure remains the same. This instruction is intended to be used in conjunction with the following instruction.

•••

## Read inactive elements from file

1. **inactive_file** Name of the file from which inactive element information is read.

Reads element numbers from the specified file and sets them as inactive. The input file contains a list of element numbers, one entry per line. Note that successive calls to this instruction are cumulative if different sets of inactive elements are read.

•••

### 2.5.9 Parallel Simulation

The Jacobian matrix assembly and the linear solver (BiCGStab with ILU0 preconditioner) in **HydroGeoSphere** are parallelized via multi-threaded, shared memory parallelism with concurrency managed by the OpenMP API. Parallelization of the linear solver is achieved by a domain decomposition approach in which the model domain is divided into node blocks of roughly equal size and each node block is assigned to a separate thread. Figure 2.5 illustrates the subdomain partitioning for a simple structured 2-D mesh and Figure 2.6 illustrates the subdomain partitioning for a large-scale model of Canada. Due to the overhead of thread synchronization, excessive partitioning may introduce computational costs that outweigh the benefit of parallelization. Hence, it is important to strike a balance when choosing the number of subdomains. For optimal parallel efficiency we recommend approximately $10^5$ unknowns per subdomain. Please note that excessive partitioning may cause the domain decomposition partitioning to fail, in which case `phgs.exe` will automatically reduce the number of subdomains. We further recommend that unless your model is very large, you first experiment without parallelization, which is the default run mode. For additional detailed information on **HydroGeoSphere** parallelization and parallel performance we recommand the article by Hwang et al. (2014).

Parallel execution in **HydroGeoSphere** is activated by configuring the `parallelindx.dat` file prior to running the simulation. If the `parallelindx.dat` file does not exist, then `phgs.exe` will create it when it is launched. This file tells `phgs.exe` how many processors to use for the simulation. By default, this file is created assuming that the simulation is to be run in serial mode, i.e., one processor.

```
__Number_of_CPU
           1
__Writing_Output_Time
  -1.00000000000000
__Simulation_Restart
           1
```

To increase the level of parallelization, change the value of `__Number_of_CPU`, which must be an even number and ideally should be a power of two. When setting the number of CPUs/subdomains it is important to make sure you do not exceed the number of processors available on your machine. In general, we recommend that you use up to two fewer than the total number available. For example, if your machine has eight processors, we recommend that you use up to six if you plan on using the machine for other tasks.

The following example shows how the `parallelindx.dat` file would be set up to use six processors for a simulation.

```
__Number_of_CPU
           6
__Writing_Output_Time
  -1.00000000000000
__Simulation_Restart
           1
```

You do not have to wait for `phgs.exe` to generate the `parallelindx.dat` file each time you run a simulation. You can copy the file from a previous simulation to your current model folder. Changing the `parallelindx.dat` file while a simulation is running will not affect

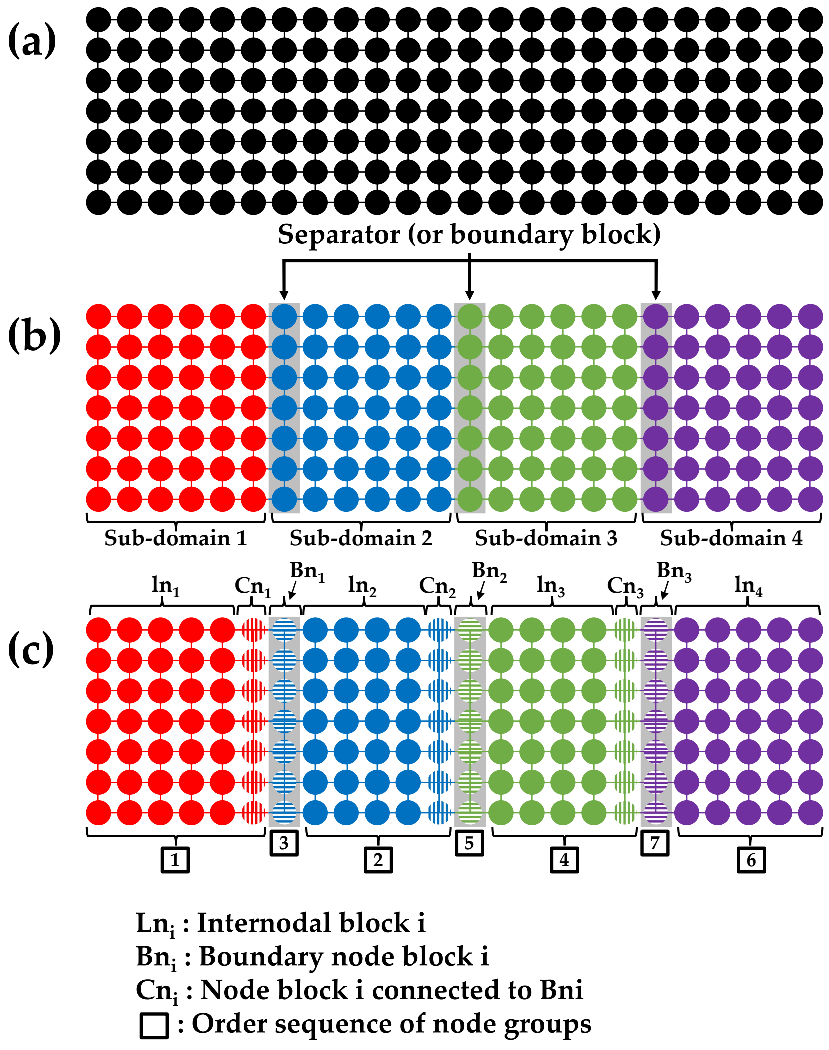

(a)
Separator (or boundary block)
(b)
Sub-domain 1
Sub-domain 2
Sub-domain 3
Sub-domain 4
(c)
In1
Cn1
Bn1
In2
Cn2
Bn2
In3
Cn3
Bn3
In4
1
3
2
5
4
7
6
Ln_i : Internodal block i
Bn_i : Boundary node block i
Cn_i : Node block i connected to Bni
□ : Order sequence of node groups

Figure 2.5: Partitioning of structured 2-D mesh into four subdomains (adapted from Hwang et al. (2014, Figure 7)).

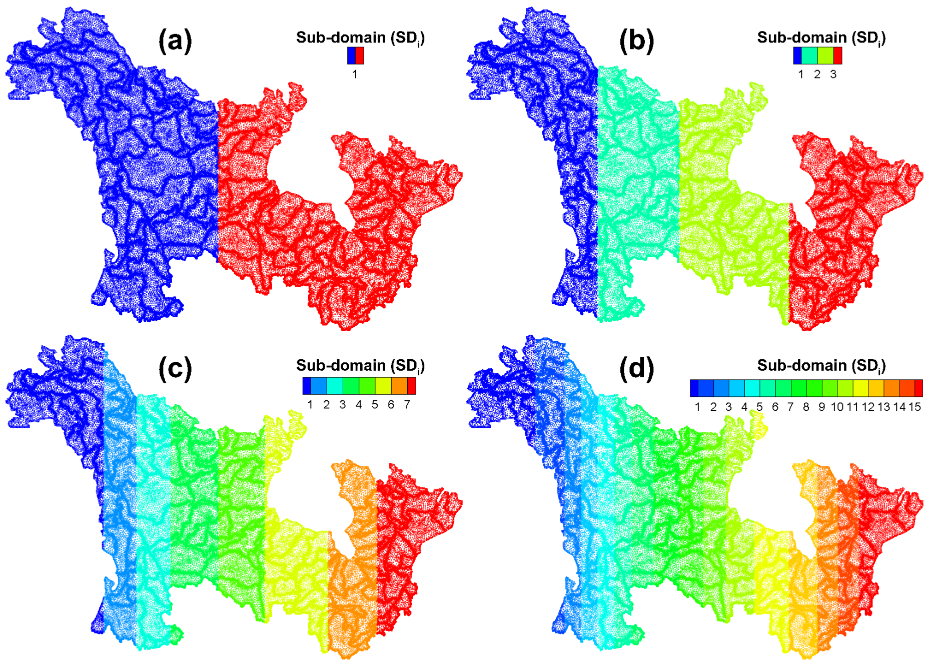


Figure 2.6: Partitioning of large-scale Canada model into: (a) 2 subdomain, (b) 4 subdomains, (c) 8 subdomains, (d) 16 subdomains Hwang et al. (2014, Figure 13).

the number of processors being used. To change the level of parallelization it is necessary to stop and restart the simulation. Note that older versions of the `parallelindx.dat` file may contain the directives `__Num_Domain_Partitiong`, `__Solver_Type`, which are no longer supported, and the directive `__Coloring_Input`, which is deprecated but remains operational.

The remaining directives in the `parallelindx.dat` file have the following effect. The directive `__Writing_Output_Time` may be used to limit the total simulation wall time to a user specified amount of time in seconds. Specifying any negative value disables this behavior, which is the default setting. Finally, the directive `__Simulation_Restart` can be used to signal a simulation restart as discussed in Section 2.5.10.

### 2.5.10 Restarting a Simulation

In the event that a **HydroGeoSphere** simulation fails or is terminated by the user, it can be restarted at a point near to where it stopped. Restarting is available for most simulation settings including transient flow and solute transport with steady-state/transient flow, but is not applicable to steady-state flow simulations. To restart a simulation, simply open the `parallelindx.dat` file and set the *restart index*, i.e., the number under the heading

```
__Simulation_Restart
```

to any integer greater then one. Then start your simulation again, without running **grok**, and it will continue from a point near to where it stopped. When restarting, head, concentration (when running transport), and other state data are read from the binary file `restart_state.dat` and used as initial conditions for the restarted simulation. The initial time and timestep of the restarted simulation as well as some other timestepping details are read from the ASCII file `restart_file_info.dat`. These restart files are generated and updated automatically by **HydroGeoSphere** during a simulation.

Each time **HydroGeoSphere** updates the restart files it first renames them to `restart_file_info_prev.dat` and `restart_state_prev.dat`. That way, if something goes wrong during their update, then only current progress is lost. If restarting a simulation fails during reading of the restart files, then the likely culprit is file corruption. If that happens, try renaming the old files (`restart_file_info_prev.dat` and `restart_state_prev.dat`) to `restart_file_info.dat` and `restart_state.dat`, respectively. You should then be able to restart **HydroGeoSphere** from an uncorrupted initial state.

By default, restarting a simulation causes output to be appended to existing output files. You can instead cause new output files to be generated with the restart index appended to their name by editing the file `restart_file_info.dat`. For example, if the restart index was set to 2, then mass balance information for the flow solution of the restarted simulation would be written to the file *prefix*`o.water_balance.0002.dat`. Simply set the value under the heading

```
__append_to_output_files(F=false,T=true)
```

to `F` for false.

You can control the frequency at which restart data is updated via the following **grok** command.

## Restart file save interval

1. **restart_interval** Time interval in seconds for saving restart information.

This command causes restart information to be saved every **restart_interval** seconds as measured by the simulation wall clock time. Note that setting **restart_interval** to any value less than or equal to zero will cause restart information to be saved after every timestep, which is the default setting.

●●●

In certain situations you may want to disable updating of the restart files during a simulation. You can do so via the command Restart write off (restart files are always updated at the end of a successful simulation regardless of this command). Use this command with caution, however, since if your simulation does not complete successfully you will be unable to restart it via the restart feature.

## Restart write off

Disables updating of the restart files `restart_state.dat` and `restart_file_info.dat`. By default these files are updated automatically during a simulation.

●●●

Note that when restarting a simulation it is very important that you do not change your model setup, i.e., your *prefix*`.grok` file followed by running **grok**. If you do need to change your model setup, then it is recommended that you start a new simulation.

## 2.6 Initial Conditions

### 2.6.1 Subsurface Flow

Initial heads should be given for both steady-state and transient problems since the iterative solver uses them as a starting point in achieving a solution. These head values can be either assigned or read from a file by one of the following commands.

#### Initial head

1. **hval** Initial head [L].

Chosen nodes in the currently active domain (see Section 2.8.1) are assigned the initial head value **hval**.

●●●

#### Initial head surface elevation

All nodes in the currently active domain (see Section 2.8.1) are assigned an initial head value equal to the surface elevation at the same $xy$-location.

●●●

#### Initial head surface elevation for chosen nodes

Chosen nodes in the currently active domain (see Section 2.8.1) are assigned an initial head value equal to the surface elevation at the same $xy$-location.

●●●

#### Initial head from depth-saturation table

1. **depth(i), saturation(i)...end** Depth [L] and saturation [-] list. Note that paired values of depth and saturation should be entered from smallest to largest depth.

Chosen nodes in the currently active domain (porous media, dual continua, or fracture; see Section 2.8.1) are first assigned a saturation that is interpolated from the depth-saturation table. The nodal depth is calculated relative to the node on ground surface at the same $xy$-location. The computed saturation is then converted to an initial head value via the constitutive relationships (see Section 2.8.3) for the zone number of the element containing the chosen node. In cases where a node is shared by two elements with different zone numbers, the zone number of the lowest numbered element is used.

●●●

## Initial head from file

1. **filename** Name of the file that contains the initial head [L] data.

All nodes in the currently active domain (porous media, dual continua, surface flow, or fracture; see Section 2.8.1) are assigned an initial head value that is read from a free-format ASCII file. The heads must be listed in the file in order from node 1 to node $N$, where $N$ is the number of nodes in the active domain. Each line may contain one or more values.

●●●

## Initial head from output file

1. **filename** Name of the file that contains the initial head [L] data.

All nodes in the currently active domain (see Section 2.8.1) are assigned an initial head value which is read from a previously generated output file named *prefix*`o.head_`*domain*`.`*suffix*. Here, *domain* specifies the active domain (*pm*, *dual*, *olf*, *frac*, *well*, *tile*, *chan*) and *suffix* is a zero-padded integer identifying the output file. This can be useful if you want to use separate solutions of groundwater and surface flow to start a coupled surface-subsurface simulation.

●●●

## Initial head subsurface from surface output file

1. **filename** Name of the file that contains the initial surface domain head [L] data.

All nodes in the currently active domain (porous media or dual continua; see Section 2.8.1) are assigned an initial head value equal to the surface domain head at the same $xy$-location. The surface heads are read from a previously generated output file named *prefix*`o.head_olf.`*suffix*, where suffix is a zero-padded integer identifying the output file.

●●●

## Function x initial head

1. **x1, h1** $x$-coordinate [L] and initial head [L] value for the first point.
2. **x2, h2** $x$-coordinate [L] and initial head [L] value for the second point.

Chosen nodes in the currently active domain (porous media, dual continua, surface flow, or fracture; see Section 2.8.1) are assigned initial head values, as a linear function of their $x$-coordinate.

●●●

## Function y initial head

The same as command Function x initial head but as a function of $y$.

●●●

## Function z initial head

The same as command Function x initial head but as a function of $z$.

●●●

## Initial head from raster

1. **rasterfile, (band)** Name of the raster file that contains the initial head [L] data and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).

Chosen nodes in the currently active domain (see Section 2.8.1) are assigned initial head values interpolated from the specified raster band.

●●●

## Initial head depth to water table

1. **depth2wtable** Depth [L] of water table below ground surface.

For each node in the currently active domain (see Section 2.8.1), a value for the initial head will be calculated based on the assigned depth to water table.

●●●

## Function surface elevation initial head

1. **z0, zscale** Reference elevation [L] and scaling factor [-].

Chosen nodes in the currently active domain (see Section 2.8.1), are assgined an initial head $h$ as a function of the ground surface elevation ($z_{surf}$) according to

$$h = \mathbf{z0} + \mathbf{zscale} \cdot (z_{surf} - \mathbf{z0})$$

●●●

## Initial head from xyh file

1. **max_dist** Maximum distance [L] threshold for determining nearest points.
2. **filename** Name of the file that contains the initial head data.

Chosen nodes in the currently active domain (porous media, dual continua, or surface flow; Section 2.8.1), are assigned an intial head that is read from a free-format ASCII file. The first line of the ASCII file contains the number of data points. Subsequent lines contain three floating point values for the $xy$-coordinates [L] and head [L] value, i.e., $x\ y\ h(x, y)$. For each chosen node, the head of the nearest point in the file is assigned if the distance between the node and that point is less than **max_dist**.

●●●

## Initial head from porous medium

For all nodes in the currently active domain (see Section 2.8.1), the initial head is set to be the same as the initial head specified in the porous medium domain.

●●●

## Compute velocity field from head

1. **v_head_file** Name of the file that contains the head [L] values of a previous flow solution.

A flow solution is not performed. Instead, the velocity field is computed from the previous flow solution, a steady-state flow field is assumed, and the transport solution proceeds. The head values are read from a previously generated output file named *prefix*`o.hen`.

●●●

## Compute velocity field from head and conc

1. **v_head_file** Name of the file that contains the head [L] values of a previous flow solution.

2. **v_conc_file** Name of the file that contains the conentration [M $L^{-3}$] values of a previous transport solution.
3. **vrhomax, vcmax** Assumed maximum density [M $L^{-3}$] and concentration [M $L^{-3}$].

A flow solution is not performed. Instead, the velocity field is computed from the previous flow solution and transport solution, a steady-state flow field is assumed, and the density-dependent transport solution proceeds. The head values are read from a previously generated output file named *prefix*`o.hen`. The concentration values are read from a previously generated output file named *prefix*`o.cen`.

●●●

#### 2.6.1.1 Regional Model

The following commands can be used to define a regional model and to map various quantities from that model to the local model. Note that only one regional model can be defined in **grok** at a time.

### Regional model

1. **prefix_regional** Prefix of the regional simulation model.

This command defines a new regional model. It reads the porous medium mesh files **prefix_regional**`o.coordinates_pm` and **prefix_regional**`o.elements_pm`. If the regional model includes the overland flow or ET domains, then it also reads the corresponding mesh files for those domains.

●●●

### Map zone numbers from regional model

Maps porous media, overland flow, or ET domain zone numbers from a regional model based on the location of each element/face centroid within the regional model. This command must come after a regional model has been defined via the Regional model command.

●●●

### Map zone numbers from regional model by layer

1. **i, regional_model_layer(i), i=1,nlayers_local** Local model layer number and the mapped from regional model layer number for each layer in the local model.

Maps porous media domain zone numbers on a per layer basis from a regional model based on the location of each each element centroid within the regional model. This command must come after a regional model has been defined via the Regional model command. The command may be issued as follows:

```
Map zone numbers from regional model by layer
1 1
2 1
3 2
4 3
5 4
```

In this example, the local model has five layers. Each row specifies a mapping for a single layer in the local model. For example, the second row indicates that values in layer two of the local model are mapped from values in layer one of the regional model. Note that a mapping must be provided for each layer in the local model, but not all layers in the regional model may be mapped from (e.g., the regional model in the above example could have more than four layers).

●●●

## Map zone numbers from regional model by volume

1. **nmc** Number of Monte Carlo samples to compute volumes of intersection.

Maps porous media domain zone numbers for chosen elements from a regional model. Each chosen element is assigned the regional model zone number of the zone that has the maximum volume of intersection with that element. This command must come after a regional model has been defined via the Regional model command. Three-dimensional volumes of intersection between local and regional model elements are approximated via Monte Carlo simulation, where we recommend that the number of Monte Carlo samples (**nmc**) be set to at least 1000 and ideally 10 000. Recall that the approximation error for the volume computation scales roughly as $1/\sqrt{\mathbf{nmc}}$. Please keep in mind that this command is computationally intensive and may take some time to complete.

●●●

## Initial head from regional output file

1. **filename** Name of the head [L] output file from the regional simulation.

Chosen nodes in the currently active domain (porous media or surface flow; see Section 2.8.1) are assigned initial head values from a previously generated regional model head output file. This command must come after a regional model has been defined via the Regional model command.

●●●

## Initial head from regional output file with depth scaling

1. **filename** Name of the head [L] output file from the regional simulation.
2. **dscale** Depth scale factor [-] applied to the surface domain (must be strictly positive).

Chosen nodes in the surface flow domain are assigned initial head values from a previously generated regional model surface domain head output file. In contrast to the command Initial head from regional output file, initial head values are computed with the depth scaled by **dscale**. This command must come after a regional model has been defined via the Regional model command.

●●●

## Initial concentration from regional output file

1. **filename** Name of the concentration [M $L^{-3}$] output file from the regional simulation.

Chosen nodes in the currently active domain (porous media or surface flow; see Section 2.8.1) are assigned initial concentration values, for the first solute only, from a previously generated regional model concentration output file. This command must come after a regional model has been defined via the Regional model command.

●●●

## Map K from regional model list file

1. **filename** Name of the file that contains the variable hydraulic conductivity [L $T^{-1}$] data.

Maps variable hydraulic conductivity from a regional model defined by the Regional model command for the current set of chosen elements. The format of the input file is the same as that for the command Read elemental K from file; see Section 2.8.2.1. This command must come after a regional model has been defined via the Regional model command.

●●●

## Map K from regional model list file by layer

1. **filename** Name of the file that contains the variable hydraulic conductivity [L $T^{-1}$] data. The format of the input file is the same as that for the command Read elemental k from file; see Section 2.8.2.1.
2. **i, regional_model_layer(i), i=1,nlayers_local** Local model layer number and the mapped from regional model layer number for each layer in the local model.

Maps variable hydraulic conductivity on a per layer basis from a regional model defined by the Regional model command for the current set of chosen elements. The format of the input file is the same as that for the command Read elemental K from file; see Section 2.8.2.1. This command must come after a regional model has been defined via the Regional model command. The command may be issued as follows:

```
Map K from regional model list file by layer
input_k_filename
1 1
2 1
3 2
4 3
5 4
```

In this example the local model has five layers. Each row specifies a mapping for a single layer in the local model. For example, the second row indicates that values in layer two of the local model are mapped from values in layer one of the regional model. Note that a mapping must be provided for each layer in the local model, but not all layers in the regional model may be mapped from (e.g., the regional model in the above example could have more than four layers).

●●●

### 2.6.2 Surface Flow

#### Initial water depth

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

Chosen nodes in the surface flow domain are assigned an initial depth value.

●●●

#### Initial water depth from file

1. **filename** Name of the file that contains the initial water depth [L] data.

This command assumes that the surface flow domain covers the entire top of the 3-D domain and contains $N$ nodes, where $N$ is the number of nodes in the 2-D slice. All surface flow domain nodes (see Section 2.8.1) are assigned an initial water depth value that is read from a free-format ASCII file. The depth values must be listed in the file in order from node 1 to node $N$.

●●●

### 2.6.3 Transport

The instructions given here apply to all solutes already defined according to the instructions given in Section 2.6.3.1.

Initial concentrations for all solutes in all domains, for the immobile zone when the dual porosity option is activated, and for colloids in the liquid phase and on the solid phase, are by default equal to zero unless one of the following instructions are included.

#### Initial concentration

1. **conc(i), i=1,nspeciesmob** Initial concentration [M $L^{-3}$] for each solute.

For each solute, chosen nodes in the currently active domain (porous media, dual continua, surface flow, or fracture; see Section 2.8.1) are assigned the specified initial concentration value.

•••

#### Initial concentration from file

1. **filename** Name of the file that contains the initial concentration [M $L^{-3}$] data.

For each solute, all nodes in the currently active domain (porous media or dual continua; see Section 2.8.1) are assigned an initial concentration, which is read from a free-format ASCII file. For each solute, the concentrations must be listed in the file in order from node 1 to node $N$, where $N$ is the number of nodes in the active domain. Each line may contain one or more values. For example, if there are two solutes and three nodes in the domain, then a valid file format would be

```
c_11
c_21
c_31
c_12
c_22
c_32
```

•••

#### Initial concentration from output file

1. **filename(i), i=1,nspeciesmob** Name of the file that contains the initial concentration [M $L^{-3}$] data for each solute.

For each solute, all nodes in the currently active domain (porous media, dual continua, surface flow, or fracture; see Section 2.8.1) are assigned initial concentration values from a previously generated output file named *prefix*`o.conc_`*domain*`.`*species*`.`*suffix*. Here, *domain* specifies the active domain (*pm*, *dual*, *olf*, or *frac*), *species* is the name of the solute, and *suffix* is a zero-padded integer identifying the output file.

●●●

## Initial concentration from xyc file

1. **max_dist** Maximum search distance [L].
2. **filename** Name of the file that contains the initial concentration data.

Chosen nodes in the currently active domain (porous media, dual continua, or surface flow; see Section 2.8.1) are assigned an initial concentration, for the first solute only, which is read from a free-format ASCII file. The first line of the ASCII file contains the number of data points. Each subsequent line contains three floating point values separated by spaces for the $xy$-coordinates [L] and concentration [M L$^{-3}$] value, i.e., $x\ y\ C(x,y)$. For each chosen node, the concentration of the nearest point in the file is assigned if the distance between the node and that point is less than **max_dist**.

●●●

## Z function initial concentration

1. **z1, conc1(i), i=1,nspeciesmob** First elevation [L] and concentration [M L$^{-3}$] for each solute.
2. **z2, conc2(i), i=1,nspeciesmob** Second elevation [L] and concentration [M L$^{-3}$] for each solute.

For each solute, chosen nodes in the currently active domain (porous media, dual continua, or surface flow; see Section 2.8.1) are assigned an initial concentration as a linear function of elevation. For example, if three solutes are defined then a valid input format is

```
z1 conc_11 conc_12 conc_13
z2 conc_21 conc_22 conc_23
```

●●●

## Specified concentration from initial concentration

For each solute, a specified concentration boundary condition is applied to all chosen nodes in the currently active domain (porous media, dual continua, surface flow, or fracture; see Section 2.8.1). The specified concentration values are set equal to the initial concentration values.

●●●

## Initial concentration for zones

1. **izone(i), czone(i)...end** Zone number and initial concentration [M $L^{-3}$].

Porous media nodes that belong to elements whose zone number is in the input list are assigned the corresponding initial concentration value, for the first solute only. All other nodes are assigned an initial concentration of zero, for the first solute only. If a node straddles the boundary between one or more zones, then the average initial concentration value is assigned.

●●●

## Initial immobile concentration from output file

1. **filename** Name of the file that contains the initial immobile zone concentration [M $L^{-3}$] data.

Porous media nodes are assigned an immobile zone initial concentration, for the first solute only, from a previously generated output file named *prefix*`o.iconc_pm.`*species*`.`*suffix*. Here, *species* is the name of the solute and *suffix* is a zero-padded integer identifying the output file. This command ensures that the mobile and immobile zones will be in equilibrium at the start of a simulation if the same concentration output file is used to define the initial conditions.

●●●

## Initial immobile zone concentration

1. **conc** Initial concentration [M $L^{-3}$].

Porous media nodes are assigned the immobile zone initial concentration **conc** for the first solute only.

●●●

## Initial immobile zone concentration from file

1. **filename** Name of the file that contains the initial immobile zone concentration [M $L^{-3}$] data.

Porous media nodes are assigned an immobile zone initial concentration, for the first solute only, which is read from a free-format ASCII file. The concentrations must be listed in the file in order from node 1 to node $N$, where $N$ is the total number of nodes in the active domain. Each line may contain one or more values.

● ● ●

## Initial concentration solid phase

1. **conc_solid(i), i=1,nspeciesmob** Initial solid phase concentration [$N_c$ $M^{-1}$] for each solute.

The instruction is only applicable to the porous media or dual continua domains (see Section 2.8.1). For each solute, chosen nodes in the currently active domain are assigned the specified initial solid phase concentration value. Colloid transport parameters must have been assigned to at least one solute. For solutes that have not been assigned colloid transport parameters, a value must still be specified but it will not be used for any computation.

● ● ●

## Initial concentration solid phase from file

1. **filename** Name of the file that contains the initial solid phase concentration [$N_c$ $M^{-1}$] data.

The instruction is only applicable to the porous media or dual continua domains (see Section 2.8.1). For each solute, all nodes in the currently active domain are assigned an initial solid phase concentration, which is read from a free-format ASCII file. For each solute, the concentrations must be listed in the file in order from node 1 to node $N$, where $N$ is the number of nodes in the active domain. Each line may contain one or more values. For example, if there are two solutes and three nodes in the domain, then a valid file format would be

```
c_11
c_21
c_31
c_12
c_22
c_32
```

Colloid transport parameters must have been assigned to at least one solute. For solutes that have not been assigned colloid transport parameters, initial concentrations will still be read from file but they will not be used for any computation.

● ● ●

## Initial concentration solid phase from output file

1. **filename(i), i=1,nspeciesmob** Name of the file that contains the initial solid concentration [$N_c$ $M^{-1}$] data for each solute.

The instruction is only applicable to the porous media or dual continua domains (see Section 2.8.1). For each solute, all nodes in the currently active domain are assigned initial solid phase concentration values from a previously generated output file named *prefix*-`o.conc_solid_`*domain*`.`*species*`.`*suffix*. Here, *domain* specifies the active domain (*pm* or *dual*), *species* is the name of the solute, and *suffix* is a zero-padded integer identifying the output file. Colloid transport parameters must have been assigned to at least one solute. For solutes that have not been assigned colloid transport parameters, initial concentrations will still be read from file but they will not be used for any computation.

• • •

## Use Pitzer model

Causes fluid density to be calculated from species concentrations via Pitzer's ion interaction model (Pitzer, 1973; Pitzer and Mayorga, 1973; Pitzer, 1991). Note that this method can be very time-consuming because fluid density is calculated by iterating between density and molality. The default is a faster and non-iterative empirical approach.

• • •

The following commands define time and depth dependent 1-D temperature profiles for the bulk porous medium and bulk dual continuum domains (Equation 2.115). They are intended to be used in conjunction with the commands Nonlinear decay with temperature, Zoned nonlinear decay with temperature, Dual nonlinear decay with temperature, and Zoned dual nonlinear decay with temperature that form part of the solute definition.

## Solute 1d temperature profile...End

Defines a depth-dependent 1-D temperature profile for the bulk porous medium. Commands are read until an End command is encountered. Note that all temperatures must be specified in degrees Celsius. For example:

```
solute 1d temperature profile
  surface temperature
      0           -7.0
      2592000     -6.1
      5184000     -0.9
      7776000      6.2
      10368000    12.9
      12960000    18.0
```

```
        15552000   20.2
        18144000   19.3
        20736000   14.8
        23328000    8.6
        25920000    2.4
        28512000   -4.0
        31104000   -7.0
    end

    thermal diffusivity
    2.0e-7

    background temperature
    6.0

    maximum depth
    2.0

    integration convergence tolerance
    0.01

    integration maximum iterations
    20

    number of points
    500

    integration memory length
    15552000.0
  end
```

●●●

## Solute dual 1d temperature profile...End

Defines a depth-dependent 1-D temperature profile for the bulk dual continuum. Commands are read until an End command is encountered. Note that all temperatures must be specified in degrees Celsius. See the description of the command Solute 1d temperature profile...End for an example of its usage.

●●●

The commands Solute 1d temperature profile...End and Solute dual 1d temperature profile...End are composed of the following subcommands:

## Surface temperature

1. **time(i), temp(i)...end** Time [T] and surface temperature [°C] value list.

At time **time(i)** the value **temp(i)** is applied and maintained until time **time(i + 1)**. The last value entered in the list will be applied until the end of the simulation.

●●●

## Thermal diffusivity

1. **thermal_diff** Thermal diffusivity [$L^2$ $T^{-1}$]. Default value of $2 \times 10^{-7}$ $m^2$ $s^{-1}$.

●●●

## Background temperature

1. **background_temp** Background temperature [°C].

●●●

## Maximum depth

1. **max_depth** Maximum depth [L] for which the temperature profile is defined.

Note that the temperature beyond the maximum depth is constant and is equal to the temperature at the maximum depth.

●●●

## Integration convergence tolerance

1. **tol** Convergence tolerance [$T^{-1/2}$ °C]. Default value of 0.01 $s^{-1/2}$ °C.

Controls the approximation quality of the integral that defines the porous medium temperature. If $I_n$ is the approximation of the integral on $n$ points and $I_{2n}$ is the approximation of the integral on $2n$ points, then convergence of this approximation is defined by

$$|I_{2n} - I_n| \leq \textbf{tol},$$

where $n = 2^k n_0$ for $k = 0, 1, 2, \ldots$.

●●●

## Integration maximum iterations

1. **maxits** Maximum number of iterations for integration convergence. Default value of 20.

Terminates the approximation of the integral that defines the porous medium temperature after **maxits** iterations and issues a warning message to the screen and the *prefix*`o.lst` file.

•••

## Integration memory length

1. **memory_length** Memory length for convolution integral [T]. Default value of 15 552 000 s (6 months).

The memory length of the convolution integral can be set to control the amount of thermal memory in the system. After a certain amount of time relative to the past, there should be no influence on the present value of the porous medium temperature. In practice, the memory length of the convolution integral should be set to half the surface temperature cycle length. For example, if seasonally varying ground surface temperature is cyclic over a period of a year, then an appropriate memory length is 6 months (in consistent time units). If a diurnal temperature cycle was being simulated, then an appropriate memory length would be 12 hours.

•••

## Number of points

1. **num** Number of values in depth-dependent temperature profile. Default value of 500.

At a given time $t$, a depth-dependent temperature profile is defined as a lookup table over a range of **num** uniformly spaced depth values between zero and the maximum depth defined by the command Maximum depth. For a given depth value $z$ in this range, the temperature is interpolated, via linear interpolation, from the endpoints of the bin that contains $z$. For any value of $z$ that falls outside this range, the temperature is treated as constant and equal to the temperature of the endpoint nearest to $z$.

•••

#### 2.6.3.1 Solute Definition

These instructions can be used to add a new solute (i.e., species) to the system. **HydroGeoSphere** is able to handle more than one solute per simulation, and straight and branching decay chains are also supported. An example of a straight decay chain is the following system:

$$\text{Uranium}^{234} \rightarrow \text{Thorium}^{230} \rightarrow \text{Radium}^{226}$$

which indicates that the decay of the radioactive isotope Uranium$^{234}$ produces the daughter product Thorium$^{230}$, which in turn decays to form Radium$^{226}$. For an example of a straight decay chain see Section 1.5.1 in the Verification Manual. Branching decay chains can have a single isotope which decays into one or more daughter products, or daughter products which have one or more parents.

Note that a solute can have different values for the decay constant and distribution coefficient (retardation factor for fractured media) in porous, dual or fractured media or from zone to zone in a single medium.

## Solute...End

Causes **grok** to begin reading a group of solute definition instructions until it encounters an End instruction.

•••

The available instructions are:

## Name

1. **spname** Solute name, up to 40 characters.

Changes the solute name, which defaults to "Species n", where n is the current solute number.

•••

## Free-solution diffusion coefficient

1. **diffrac** Free-solution diffusion coefficient [$L^2$ $T^{-1}$].

Assigns a new value for the free-solution diffusion coefficient, $D_{free}$ (Equation 2.122). The default value is $1.0 \times 10^{-9}$ m$^2$ s$^{-1}$.

•••

## Parents

1. **npa** Number of parent species for the current species.

2. **kparen(i), aparen(i), i=1,npa** Parent species number and the mass fraction [-].

Assigns a value for the number of parent species, which defaults to 0. The mass fraction is a number between 0 and 1 which defines how much of the parent species transforms into the daughter species (i.e., the current species).

•••

The following parameters affect porous media solute properties:

## Isotropic effective diffusion coefficient

1. **coeff** Effective diffusion coefficient [$L^2$ $T^{-1}$].

Assigns an isotropic effective diffusion coefficient, $\tau D_{free}$ in Equation 2.122. A uniform value is assigned for all porous media zones in the domain. This command overrides the command Free-solution diffusion coefficient. If this command is not specified, the effective diffusion coefficient is defined in terms of the free-solution diffusion coefficient, $D_{free}$, given by the command Free-solution diffusion coefficient.

•••

## Zoned isotropic effective diffusion coefficient

1. **coeff(j), j=1,nzones** Effective diffusion coefficient [$L^2$ $T^{-1}$] for each porous media zone **j**.

Assigns isotropic effective diffusion coefficients, $\tau D_{free}$ in Equation 2.122. A unique value is assigned to each porous media zone in the domain. This command overrides the command Free-solution diffusion coefficient. If this command is not specified, the effective diffusion coefficient is defined in terms of the free-solution diffusion coefficient, $D_{free}$, given by the command Free-solution diffusion coefficient.

•••

## Anisotropic effective diffusion coefficient

1. **coeff_xx, coeff_yy, coeff_zz** Effective diffusion coefficients [$L^2$ $T^{-1}$] in the $x$-, $y$-, and $z$-directions.

Assigns an anisotropic effective diffusion coefficient, $\tau D_{free}$ in Equation 2.122. A uniform value is assigned for all porous media zones in the domain. This command overrides the command Free-solution diffusion coefficient. If this command is not specified, the effective diffusion coefficient is defined in terms of the free-solution diffusion coefficient, $D_{free}$, given by the command Free-solution diffusion coefficient.

•••

## Zoned anisotropic effective diffusion coefficient

1. **coeff_xx(j), coeff_yy(j), coeff_zz(j), j=1,nzones** Effective diffusion coefficients [L$^2$ T$^{-1}$] in the $x$-, $y$-, and $z$-directions for each porous media zone **j**.

Assigns anisotropic effective diffusion coefficients, $\tau D_{free}$ in Equation 2.122. A unique value is assigned to each porous media zone in the domain. This command overrides the command Free-solution diffusion coefficient. If this command is not specified, the effective diffusion coefficient is defined in terms of the free-solution diffusion coefficient, $D_{free}$, given by the command Free-solution diffusion coefficient.

•••

## Decay constant

1. **clambda** First-order decay constant [T$^{-1}$].

Assigns a uniform value for the solute first-order decay constant, $\lambda$ (Equation 2.120), for all porous media zones in the domain. The default value is 0.0 (no decay).

•••

## Zoned decay constant

1. **clambda(j), j=1,nzones** First-order decay constant [T$^{-1}$] for each porous media zone **j**.

Assigns a unique value for the solute first-order decay constant, $\lambda$ (Equation 2.120), to each porous media zone in the domain. The default value is 0.0 (no decay).

•••

## Nonlinear decay with saturation

1. **coeff** Nonlinear decay coefficient $\alpha$ [T$^{-1}$].

2. **rsat** Nonlinear decay reference saturation $S_r$ [-].

3. **shape** Nonlinear decay shape parameter $\beta$ [-].

Causes the first-order decay constant, $\lambda$ (Equation 2.120), across all porous medium zones in the domain to be computed according to the nonlinear decay equation

$$\lambda = \alpha \cdot \min\left[1, \left(\frac{S_w}{S_r}\right)^{\beta}\right],$$

where $S_w$ is the water saturation. This command can be combined with the commands Nonlinear decay with temperature or Zoned nonlinear decay with temperature, for the same species, in which case the decay coefficients have a multiplicative effect. Note that this command overrides the commands Decay constant and Zoned decay constant for the same species.

● ● ●

## Zoned nonlinear decay with saturation

1. **coeff(j), rsat(j), shape(j), j=1,nzones** Nonlinear decay coefficient $\alpha_j$ [T$^{-1}$], reference saturation $S_{r,j}$ [-], and shape parameter $\beta_j$ [-], respectively, for each porous medium zone **j**.

Causes the first-order decay constant, $\lambda$ (Equation 2.120), for each porous medium zone in the domain to be computed according to the nonlinear decay equation

$$\lambda = \alpha_j \cdot \min\left[1, \left(\frac{S_w}{S_{r,j}}\right)^{\beta_j}\right],$$

where $S_w$ is the water saturation and $j$ indexes the $j$th porous medium zone. This command can be combined with the commands Nonlinear decay with temperature or Zoned nonlinear decay with temperature, for the same species, in which case the decay coefficients have a multiplicative effect. Note that this command overrides the commands Decay constant and Zoned decay constant for the same species.

● ● ●

## Nonlinear decay with temperature

1. **ref_coeff** Decay coefficient at reference temperature $\alpha$ [T$^{-1}$].
2. **ref_temp** Reference temperature $T_r$ [°C].
3. **act_energy** Activation energy of the reaction $E_a$ [J mol$^{-1}$].

Computes a temperature dependent first-order decay coefficient, $\lambda$ (Equation 2.120), according to the modified Arrhenius equation

$$\lambda = \alpha \cdot \exp\left[\frac{E_a(T_b - (T_r + 273.15))}{R \cdot T_b(T_r + 273.15)}\right],$$

where $T_b$ is the temperature in Kelvin of the bulk porous medium and $R$ is the universal gas constant [J K$^{-1}$ mol$^{-1}$]. In order to use this command, the user must issue the command Solute 1d temperature profile...End once outside of the solute definition to define a depth-dependent 1-D temperature profile of the bulk porous medium. This command can be combined with the commands Nonlinear decay with saturation or Zoned nonlinear decay with saturation, for the same species, in which case the decay coefficients have a multiplicative effect. Note, however, that this command overrides the commands Decay constant and Zoned decay constant for the same species.

• • •

## Zoned nonlinear decay with temperature

1. **ref_coeff(j), j=1,nzones** Decay coefficient at reference temperature $\alpha_j$ [T$^{-1}$] for each porous medium zone **j**.
2. **ref_temp** Reference temperature $T_r$ [°C].
3. **act_energy** Activation energy of the reaction $E_a$ [J mol$^{-1}$].

Computes a temperature dependent first-order decay coefficient, $\lambda$ (Equation 2.120), according to the modified Arrhenius equation

$$\lambda = \alpha_j \cdot \exp\left[\frac{E_a(T_b - (T_r + 273.15))}{R \cdot T_b(T_r + 273.15)}\right],$$

where $T_b$ is the temperature in Kelving of the bulk porous medium and $R$ is the universal gas constant [J K$^{-1}$ mol$^{-1}$]. In order to use this command, the user must issue the command Solute 1d temperature profile...End once outside of the solute definition to define a depth-dependent 1-D temperature profile of the bulk porous medium. This command can be combined with the commands Nonlinear decay with saturation or Zoned nonlinear decay with saturation, for the same species, in which case the decay coefficients have a multiplicative effect. Note, however, that this command overrides the commands Decay constant and Zoned decay constant for the same species.

• • •

## Distribution coefficient

1. **dkd** Distribution coefficient [M$^{-1}$ L$^3$].

Assigns a uniform value for the solute distribution coefficient, $K'$ (Equation 2.121), for all porous media zones in the domain. The default value is 0.0 (no attenuation).

• • •

## Zoned distribution coefficient

1. **dkd(j), j=1,nzones** Distribution coefficient [M$^{-1}$ L$^3$] for each porous media zone **j**.

Assigns a unique value for the distribution coefficient, $K'$ (Equation 2.121), to each porous media zone in the domain. The default value is 0.0 (no attenuation).

● ● ●

## Freundlich isotherm

1. **coeff** Freundlich adsorption capacity $K_F$ [M$^{-1}$ L$^3$].
2. **rconc** Freundlich reference concentration $C_0$ [M L$^{-3}$].
3. **expon** Freundlich exponent of non-linearity $n$ [-].

The nonlinear Freundlich isotherm relates the amount of adsorbate in equilibrium, $X$, to solute concentration, $C$, via the empirical relationship

$$X = K_F \cdot C_0 \left( \frac{C}{C_0} \right)^n$$

Note that for $n = 1$, we recover the linear Freundlich isotherm. The solution distribution coefficient, $K'$ (Equation 2.121), across all porous medium zones in the domain is defined as the slope of the nonlinear Freundlich isotherm

$$K' = \frac{dX}{dC} = n \cdot K_F \left( \frac{C}{C_0} \right)^{n-1}$$

Note that this command overrides the commands Distribution coefficient and Zoned distribution coefficient for the same species.

● ● ●

## Zoned freundlich isotherm

1. **coeff(j), rconc(j), expon(j), j=1,nzones** Freundlich adsorption capacity $K_{F,j}$ [M$^{-1}$ L$^3$], reference concentration $C_{0,j}$ [M L$^{-3}$], and exponent of non-linearity $n_j$ [-], respectively, for each porous medium zone **j**.

The nonlinear Freundlich isotherm relates the amount of adsorbate in equilibrium, $X$, to solute concentration, $C$, via the empirical relationship

$$X = K_{F,j} \cdot C_{0,j} \left( \frac{C}{C_{0,j}} \right)^{n_j}$$

Note that for $n_j = 1$, we recover the linear Freundlich isotherm. The solution distribution coefficient, $K'$ (Equation 2.121), for each porous medium zone in the domain is defined as the slope of the nonlinear Freundlich isotherm

$$K' = \frac{dX}{dC} = n_j \cdot K_{F,j} \left( \frac{C}{C_{0,j}} \right)^{n_j - 1}$$

where $j$ indexes the $j$th porous medium zone. Note that this command overrides the commands Distribution coefficient and Zoned distribution coefficient for the same species.

● ● ●

## Solute uptake by plant roots

1. **partition_constant** Fraction of solute in the transpiration stream relative to the full stream (value between 0 and 1) in the porous medium domain [-]. Note that a value of one means full transpiration-stream solute uptake across the plant root depth.

Causes the uptake of solute by plant roots. By default there is no solute uptake.

● ● ●

The following parameters affect the dual media solute properties:

## Dual decay constant

1. **clambda** First-order decay constant [$T^{-1}$].

Assigns a uniform value for the solute first-order decay constant, $\lambda_d$ (Equation 2.128), for all dual continua zones in the domain. The default value is 0.0 (no decay).

● ● ●

## Zoned dual decay constant

1. **clambda(j), j=1,nzones** First-order decay constant [$T^{-1}$] for each dual continua zone **j**.

Assigns a unique value for the solute first-order decay constant, $\lambda_d$ (Equation 2.128), to each zone in the domain. The default value is 0.0 (no decay).

● ● ●

## Dual nonlinear decay with saturation

1. **coeff** Nonlinear decay coefficient $\alpha$ [$T^{-1}$].

2. **rsat** Nonlinear decay reference saturation $S_r$ [-].

3. **shape** Nonlinear decay shape parameter $\beta$ [-].

Causes the first-order decay constant, $\lambda_d$ (Equation 2.128), across all dual continua zones in the domain to be computed according to the nonlinear decay equation

$$\lambda_d = \alpha \cdot \min\left[1, \left(\frac{S_w}{S_r}\right)^{\beta}\right],$$

where $S_w$ is the water saturation. This command can be combined with the commands Dual nonlinear decay with temperature or Zoned dual nonlinear decay with temperature, for the same species, in which case the decay coefficients have a multiplicative effect. Note that this command overrides the commands Dual decay constant and Zoned dual decay constant for the same species.

● ● ●

## Zoned dual nonlinear decay with saturation

1. **coeff(j), shape(j), j=1,nzones** Nonlinear decay coefficient $\alpha_j$ [T$^{-1}$], reference saturation $S_{r,j}$ [-], and shape parameter $\beta_j$ [-], respectively, for each dual continua zone **j**.

Causes the first-order decay constant, $\lambda_d$ (Equation 2.128), for each dual continua zone in the domain to be computed according to the nonlinear decay equation

$$\lambda_d = \alpha_j \cdot \min\left[1, \left(\frac{S_w}{S_{r,j}}\right)^{\beta_j}\right],$$

where $S_w$ is the water saturation and $j$ indexes the $j$th dual continua zone. This command can be combined with the commands Dual nonlinear decay with temperature or Zoned dual nonlinear decay with temperature, for the same species, in which case the decay coefficients have a multiplicative effect. Note that this command overrides the commands Dual decay constant and Zoned dual decay constant for the same species.

● ● ●

## Dual nonlinear decay with temperature

1. **ref_coeff** Decay coefficient at reference temperature $\alpha$ [T$^{-1}$].
2. **ref_temp** Reference temperature $T_r$ [°C].
3. **act_energy** Activation energy of the reaction $E_a$ [J mol$^{-1}$].

Computes a temperature dependent first-order decay coefficient, $\lambda_d$ (Equation 2.128), according to the modified Arrhenius equation

$$\lambda_d = \alpha \cdot \exp\left[\frac{E_a(T_b - (T_r + 273.15))}{R \cdot T_b(T_r + 273.15)}\right],$$

where $T_b$ is the temperature in Kelvin of the bulk dual continuum and $R$ is the universal gas constant [J K$^{-1}$ mol$^{-1}$]. In order to use this command, the user must issue the command Solute dual 1d temperature profile...End once outside of the solute definition to define a depth-dependent 1-D temperature profile of the bulk dual continuum. This command can be combined with the commands Dual nonlinear decay with saturation or Zoned dual nonlinear decay with saturation, for the same species, in which case the decay coefficients have a multiplicative effect. Note, however, that this command overrides the commands Dual decay constant and Zoned dual decay constant for the same species.

●●●

## Zoned dual nonlinear decay with temperature

1. **ref_coeff(j), j=1,nzones** Decay coefficient at reference temperature $\alpha_j$ [T$^{-1}$] for each dual continua zone **j**.
2. **ref_temp** Reference temperature $T_r$ [°C].
3. **act_energy** Activation energy of the reaction $E_a$ [J mol$^{-1}$].

Computes a temperature dependent first-order decay coefficient, $\lambda_d$ (Equation 2.128), according to the modified Arrhenius equation

$$\lambda_d = \alpha_j \cdot \exp\left[\frac{E_a(T_b - (T_r + 273.15))}{R \cdot T_b(T_r + 273.15)}\right],$$

where $T_b$ is the temperature in Kelvin of the bulk dual continuum and $R$ is the universal gas constant [J K$^{-1}$ mol$^{-1}$]. In order to use this command, the user must issue the command Solute dual 1d temperature profile...End once outside of the solute definition to define a depth-dependent 1-D temperature profile of the bulk dual continuum. This command can be combined with the commands Dual nonlinear decay with saturation or Zoned dual nonlinear decay with saturation, for the same species, in which case the decay coefficients have a multiplicative effect. Note, however, that this command overrides the commands Dual decay constant and Zoned dual decay constant for the same species.

●●●

## Dual distribution coefficient

1. **dkd** Distribution coefficient [M$^{-1}$ L$^3$].

Assigns a uniform value for the solute distribution coefficient, $K'_d$ (Equation 2.129), for all dual continua zones in the domain. The default value is 0.0 (no attenuation).

●●●

## Zoned dual distribution coefficient

1. **dkd(j), j=1,nzones** Distribution coefficient [M$^{-1}$ L$^3$] for each dual continua zone **j**.

Assigns a unique value for the distribution coefficient, $K'_d$ (Equation 2.129), to each dual continua zone in the domain. The default value is 0.0 (no attenuation).

● ● ●

## Dual freundlich isotherm

1. **coeff** Freundlich adsorption capacity $K_F$ [M$^{-1}$ L$^3$].
2. **rconc** Freundlich reference concentration $C_0$ [M L$^{-3}$].
3. **expon** Freundlich exponent of non-linearity $n$ [-].

The nonlinear Freundlich isotherm relates the amount of adsorbate in equilibrium, $X$, to solute concentration, $C$, via the empirical relationship

$$X = K_F \cdot C_0 \left( \frac{C}{C_0} \right)^n$$

Note that for $n = 1$, we recover the linear Freundlich isotherm. The solution distribution coefficient, $K'_d$ (Equation 2.129), across all dual continua zones in the domain is defined as the slope of the nonlinear Freundlich isotherm

$$K'_d = \frac{dX}{dC} = n \cdot K_F \left( \frac{C}{C_0} \right)^{n-1}$$

Note that this command overrides the commands Dual distribution coefficient and Zoned dual distribution coefficient for the same species.

● ● ●

## Zoned dual freundlich isotherm

1. **coeff(j), rconc(j), expon(j), j=1,nzones** Freundlich adsorption capacity $K_{F,j}$ [M$^{-1}$ L$^3$], reference concentration $C_{0,j}$ [M L$^{-3}$], and exponent of non-linearity $n_j$ [-], respectively, for each dual continua zone **j**.

The nonlinear Freundlich isotherm relates the amount of adsorbate in equilibrium, $X$, to solute concentration, $C$, via the empirical relationship

$$X = K_{F,j} \cdot C_{0,j} \left( \frac{C}{C_{0,j}} \right)^{n_j}$$

Note that for $n_j = 1$, we recover the linear Freundlich isotherm. The solution distribution coefficient, $K'_d$ (Equation 2.129), for each dual continua zone in the domain is defined as the slope of the nonlinear Freundlich isotherm

$$K'_d = \frac{dX}{dC} = n_j \cdot K_{F,j} \left( \frac{C}{C_{0,j}} \right)^{n_j - 1}$$

where $j$ indexes the $j$th dual continua zone. Note that this command overrides the commands Dual distribution coefficient and Zoned dual distribution coefficient for the same species.

●●●

## Colloid transport parameters

1. **kret** First-order colloid retention rate coefficient on the solid phase [$\mathrm{T}^{-1}$].
2. **kdet** First-order colloid detachment coefficient from the solid phase [$\mathrm{T}^{-1}$].
3. **lambdas** First-order colloid decay rate in the solid phase [$\mathrm{T}^{-1}$].

Assigns uniform values for the first-order colloid retention rate coefficient on the solid phase $k_{ret}$ in Equation 2.149, the first-order colloid detachment coefficient from the solid phase $k_{det}$ in Equation 2.149, and the first-order colloid decay rate in the solid phase $\lambda_s$ in Equation 2.150 for all porous media zones in the domain. Issuing this instruction labels the solute as a colloid and activates the colloid transport option in HGS, including the presence of colloids on the solid phase. The default values are 0.0 (solute is not a colloid).

The colloid decay rate in the fluid phase, $\lambda_c$ in Equation 2.149, is assigned with the instructions presented above in this section for assigning a decay constant for a solute.

●●●

## Zoned colloid transport parameters

1. **kret(j), kdet(j), lambdas(j), j=1,nzones** First-order colloid retention rate coefficient on the solid phase [$\mathrm{T}^{-1}$], first-order colloid detachment coefficient from the solid phase [$\mathrm{T}^{-1}$] and first-order colloid decay rate in the solid phase [$\mathrm{T}^{-1}$] for each porous media zone **j**.

Assigns unique values for the first-order colloid retention rate coefficient on the solid phase, $k_{ret}$ in Equation 2.149, the first-order colloid detachment coefficient from the solid phase $k_{det}$ in Equation 2.149, and the first-order colloid decay rate in the solid phase $\lambda_s$ in Equation 2.150 to each porous media zone in the domain. Issuing this instruction labels the solute as a colloid and activates the colloid transport option in HGS, including the presence of colloids on the solid phase. The default values are 0.0 (solute is not a colloid).

The colloid decay rate in the fluid phase, $\lambda_c$ in Equation 2.149, is assigned with the instructions presented above in this section for assigning a decay constant for a solute.

• • •

## Dual colloid transport parameters

1. **kretd** First-order colloid retention rate coefficient on the solid phase [T$^{-1}$].
2. **kdetd** First-order colloid detachment coefficient from the solid phase [T$^{-1}$].
3. **lambdasd** First-order colloid decay rate in the solid phase [T$^{-1}$].
4. **omega** Colloid exchange rate in the fluid phase between porous medium and dual domains [T$^{-1}$].
5. **kt** Colloid transfer rate in the solid phase between porous medium and dual domains [T$^{-1}$].

Assigns uniform values for the first-order colloid retention rate coefficient on the solid phase, $k_{retd}$ in Equation (2.151), the first-order colloid detachment coefficient from the solid phase, $k_{detd}$ in Equation (2.151), the first-order colloid decay rate in the solid phase $\lambda_{sd}$ in Equation 2.152, the colloid exchange rate in the fluid phase between porous medium and dual, $\omega_{ex}$ in Equation (2.153), and the colloid transfer rate in the solid phase between porous medium and dual domains, $k_t$ in Equations (2.150) and (2.152), for all dual media zones in the domain. Issuing this instruction labels the solute as a colloid and activates the colloid transport option in HGS, including the presence of colloids on the solid phase. A dual continuum domain must have been specified to issue this instruction. The default values are 0.0 (solute is not a colloid).

The colloid decay rate in the fluid phase of the dual continuum, $\lambda_{cd}$ in Equation 2.151, is assigned with the instructions presented above in this section for assigning a decay constant for a solute.

• • •

## Dual zoned colloid transport parameters

1. **kretd(j), kdetd(j), lambdasd(j), omega(j), kt(j), j=1,nzones** First-order colloid retention rate coefficient on the solid phase [T$^{-1}$], first-order colloid detachment coefficient from the solid phase [T$^{-1}$], first-order colloid decay rate in the solid phase[T$^{-1}$], colloid exchange rate in the fluid phase between porous medium and dual domains [T$^{-1}$], and colloid transfer rate in the solid phase between porous medium and dual domains [T$^{-1}$] for each dual media zone **j**.

Assigns unique values for the first-order colloid retention rate coefficient on the solid phase, $k_{retd}$ in Equation (2.151), the first-order colloid detachment coefficient from the solid phase $k_{detd}$ in Equation (2.151), the first-order colloid decay rate in the solid phase $\lambda_{sd}$ in Equation 2.152, $\omega_{ex}$ in Equation (2.153), and the colloid transfer rate in the solid phase between porous medium and dual domains, $k_t$ in Equations (2.150) and (2.152), to each dual media zone in the domain. Issuing this instruction labels the solute as a colloid and activates the colloid transport option in HGS, including the presence of colloids on the solid phase. A dual continuum domain must have been specified to issue this instruction. The default values are 0.0 (solute is not a colloid).

The colloid decay rate in the fluid phase of the dual continuum, $\lambda_{cd}$ in Equation 2.151, is assigned with the instructions presented above in this section for assigning a decay constant for a solute.

• • •

The following parameters affect the fracture domain solute properties:

## Fracture decay constant

1. **clambda_f** First-order decay constant [T$^{-1}$].

Assigns a uniform value for the solute first-order decay constant, $\lambda_f$ (Equation 2.124), for all discrete fracture zones in the domain. The default value is 0.0 (no decay).

• • •

## Zoned fracture decay constant

1. **clambda_f(j), j=1,nzones** First-order decay constant [T$^{-1}$] for each discrete fracture zone **j**.

Assigns a unique value for the solute first-order decay constant, $\lambda_f$ (Equation 2.124), to each discrete fracture zone in the domain. The default value is 0.0 (no decay).

• • •

## Fracture retardation factor

1. **rfrac** Fracture retardation factor [-].

Assigns a uniform value for the fracture retardation factor, $R_f$ (Equation 2.125), for all discrete fracture zones in the domain. The default value is 1.0 (no attenuation).

●●●

## Zoned fracture retardation factor

1. **rfrac(j), j=1,nzones** Retardation factor [-] for each discrete fracture zone **j**.

Assigns a unique value for the fracture retardation factor, $R_f$ (Equation 2.125), to each discrete fracture zone in the domain. The default value is 1.0 (no attenuation).

●●●

The following parameters affect the overland domain solute properties:

## Overland decay constant

1. **clambda_o** First-order decay constant [$\mathrm{T}^{-1}$].

Assigns a uniform value for the solute first-order decay constant, $\lambda$ (Equation 2.135), for all overland flow zones in the domain. The default value is 0.0 (no decay).

●●●

## Zoned overland decay constant

1. **clambda_o(j), j=1,nzones** First-order decay constant [$\mathrm{T}^{-1}$] for each overland flow zone **j**.

Assigns a unique value for the solute first-order decay constant, $\lambda$ (Equation 2.135), to each overland flow zone in the domain. The default value is 0.0 (no decay).

●●●

## Overland retardation factor

1. **rolf** Overland flow retardation factor [-].

Assigns a uniform value for the overland retardation factor, $R_o$ (Equation 2.125), for all overland zones in the domain. The default value is 1.0 (no attenuation).

●●●

## Zoned overland retardation factor

1. **rfrac(j), j=1,nzones** Retardation factor [-] for each overland flow zone **j**.

Assigns a unique value for the overland flow retardation factor, $R_o$ (Equation 2.125), to each overland zone in the domain. The default value is 1.0 (no attenuation).

• • •

The following instructions can be used to identify certain species. This is especially important to calculate fluid density and viscosity (for variable-density transport) from individual species concentrations and temperature. Note that fluid temperature is treated as a mobile species.

## Sodium species

The presently defined species is identified as sodium, $Na^+$.

• • •

The following instructions can be used likewise to identify other species:

Potassium species
Calcium species
Magnesium species
Chloride species
Sulphate species
Hydrogencarbonate species
Carbonate species
Salt mass fraction
Temperature species

By default, no species impacts fluid density or viscosity. This default can be changed with the following instruction:

## Affects fluid properties

With this instruction, the presently defined solute has an impact on both fluid density and viscosity.

• • •

## Relative concentration

1. **cmax** Maximum relative concentration [-], corresponding to the fluid with maximum density. This instruction proved to be especially useful to simulate the lab experiments by Oswald and Kinzelbach (2004), where the concentration of the fluid with maximum density is not one.

2. **rhomax** Maximum fluid density [M $L^{-3}$] (which corresponds to the maximum solute concentration).

Assigns the maximum relative concentration (defaults to 1) and the maximum fluid density (defaults to 0), respectively.

•••

Note that instructions like Decay constant and Zoned decay constant are mutually exclusive for a given solute, and should not appear in the same Solute...End block. This restriction also applies to distribution coefficient definitions for all types of media. You can however, define a solute with decay or attenuation properties that are uniform throughout the domain while a second solute has a zoned behaviour.

Since a new species is created each time the instruction Solute...End is used, any instructions (e.g., Make fractures, Specified concentration, Specified third-type concentration, etc.) which depend on it should be placed after it in the *prefix*`.grok` file.

The following simple example shows how to define a single, conservative, non-decaying solute called 'Species 1' with the default free-solution diffusion coefficient:

```
Solute
End solute
```

An example of a more complex system with two solutes and seven material zones is shown in Figure 2.7 for the first solute, called DCB, which only decays in zone 1, and has distribution coefficients which vary from zone to zone.

```
solute
    name
    DCB

    free-solution diffusion coefficient
    3.689e-5           ! free solution diffusion coefficient (m2/d)

    zoned decay constant
    0.693          ! 1  first-order decay constant (1/d)
    0.0            ! 2
    0.0            ! 3
    0.0            ! 4
    0.0            ! 5
    0.0            ! 6
    0.0            ! 7

    zoned distribution coefficient
    0.0005             ! 1   distribution coefficient (kg/m3)
    0.0005             ! 2
    0.0005             ! 3
    0.0013             ! 4
    0.005              ! 5
```

```
        0.014               ! 6
        0.020               ! 7
    end solute
```

Figure 2.7: Definition of a parent solute with zoned properties.

Figure 2.8 shows how to define the second solute, called BAM, which is a daughter product of DCB, and does not decay. This solute has the same zoned distribution coefficients as the first solute.

```
    solute
        name
        BAM

        free-solution diffusion coefficient
        3.7295e-5           ! free solution diffusion coefficient (m2/d)

        parents
        1               ! i.e. DCB
        ! parent #          ! mass ratio
        !==========      =============
            1                 1.0

        decay constant
        0.0             ! first-order decay constant (1/d)

        zoned distribution coefficient
        0.0005              ! 1   distribution coefficient (kg/m3)
        0.0005              ! 2
        0.0005              ! 3
        0.0013              ! 4
        0.005               ! 5
        0.014               ! 6
        0.020               ! 7
    end solute
```

Figure 2.8: Definition of a daughter solute with zoned properties.

#### 2.6.3.2 Heat Transfer

To enable heat transfer, you must define a temperature species in the solute definition block via the command Temperature species. The thermal properties of the solids are specified in the material property file. The following instructions may be used to define the heat transfer solution.

## Thermal conductivity of air

1. **k_air** Thermal conductivity [W $m^{-1}$ $K^{-1}$] of the air phase.

Assigns a uniform value to the thermal conductivity of air.

● ● ●

## Specific heat capacity of air

1. **c_air** Specific heat capacity [J $kg^{-1}$ $K^{-1}$] of the air phase.

Assigns a uniform value to the specific heat capacity of air.

● ● ●

## Density of air

1. **rho_air** Density [kg $m^{-3}$] of the air phase.

Assigns a uniform value to the density of air.

● ● ●

## Thermal conductivity of water

1. **k_l** Thermal conductivity [W $m^{-1}$ $K^{-1}$] of the liquid phase.

Assigns a uniform value to the thermal conductivity of water. If this instructions is used, the thermal conductivity of water is assumed constant and equal to $k_l$. It is therefore not calculated from the water temperature, which is the default setting.

● ● ●

## Specific heat capacity of water

1. **c_l** Specific heat capacity [J $kg^{-1}$ $K^{-1}$] of the liquid phase.

Assigns a uniform value to the specific heat capacity of water. If this instructions is used, the specific heat capacity of water is assumed constant and equal to $c_l$. It is therefore not calculated from the water temperature, which is the default setting.

•••

## Mechanical heat dispersion

Causes mechanical heat dispersion to be simulated. The default is no mechanical heat dispersion.

•••

The following instruction can be used to define initial temperature profile.

## Initial temperature profile

1. **temp_top** Temperature [°C] at the top of the domain.
2. **temp_grad** Prevailing geothermal gradient [$L^{-1}$ K].

Calculates a depth profile of temperature and assigns the values to the initial temperature.

•••

The following instructions can be used to define a heat source boundary condition.

## Zero order source

1. **npanel** Number of panels in the time-variable, zero-order source function.
2. **ton(i), toff(i), prate(i,j), j=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], heat production rate [M $L^{-1}$ $T^{-3}$].

Nodes that belong to the chosen zones are assigned zero-order source boundary conditions.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term specified for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to one, **ton** to zero, and **toff** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **prate** should be included. For example, with three panels and two species the input format is

```
3 ! npanel
ton1 toff1 prate11 prate12
ton2 toff2 prate21 prate22
ton3 toff3 prate31 prate32
```

•••

## Exponential zero order source

1. **heat_q_zero** Heat production [M $L^{-1}$ $T^{-3}$] at time $t = 0$.
2. **heat_constant** Exponential function growth constant [$T^{-1}$].

Nodes in the chosen zones area assigned an exponentially decreasing zero-order heat source boundary conditions.

•••

## 2.7 Boundary Conditions

### 2.7.1 General

The boundary condition input routines have been completely rewritten in this version of **HydroGeoSphere**. The old routines were developed over the years by many different developers and for a host of reasons. There was a lack of continuity between the routines that has made it increasingly difficult to update and maintain **HydroGeoSphere**. With this in mind, a more well defined data structure has been developed for defining boundary conditions and, where possible, the functionality of the old routines has been incorporated. For the end user, this means having to learn a new approach to defining boundary conditions, but one that we hope will be more logical and therefore easier to understand and apply.

In its simplest form, a boundary condition is defined by a value that is associated with a node, for example, a specified head. In some cases, such as a well that has a variable pumping rate, the value may change temporally. If the well were turned off abruptly then started up again at a later time, the value would not be applied continuously. Some inputs are not defined by a single value, but rather by a two- or three-dimensional field. For example, rainfall may be given in the form of 2-D raster data defined in a certain region. Other inputs might be in the form of tables of values at defined locations, which is often the case when we incorporate information from another model. As you can see, we need a combination of data structure and input format that are general, yet flexible.

All flow boundary conditions require inputs for the boundary condition type, node, face, or segment set, and the time-varying inputs described in Sections 2.7.3, 2.7.4, and 2.7.5. The name, constraints, and Tecplot inputs are optional values. A general boundary condition layout is shown as the following instruction:

```
boundary condition
```

```
        type
            {bc_type}

        name
            {bc_name}   ! optional but recommended

        node set/face set/segment set
            {bc_set_name}

        time value table/time raster table/time file table
            {bc_time(i), bc_file(i)...end}
                or
            {bc_time(i), bc_raster(i)...end}
                or
            {bc_time(i), bc_file(i)...end}

        constraints/tecplot options   ! optional not required
    end
```

## Boundary condition...End

Defines a new boundary condition until it encounters an End instruction.

• • •

A boundary condition may be assigned a user specified name via the following instruction.

## Name

1. **bc_name** Name of boundary condition, at most 40 characters.

Assigns the name **bc_name** to a boundary condition. By default, if no name is given a unique one is generated. Note that boundary condition names must be unique (case-insensitive comparison) and should not contain any whitespace characters.

• • •

### 2.7.2 Set Creation

The generic boundary condition format requires that either a node set, face set, or segment set be created. Once you have selected nodes you can create the appropriate sets by issuing the following commands:

## Create node set

1. **set_name** Name of the node set, up to 40 characters.

Creates a node set with the given name from the currently selected nodes. The node set is written to the ASCII output file *prefix*`o.node_set.`**set_name**.

●●●

## Create node set from shp

1. **filename** Name of the shapefile without the file extension.
2. **name_attribute** Field name (up to 11 characters) used to assign the node set name, which must be written exactly as in the `.dbf` file (case sensitive).
3. **bot_sheet_attribute** Field name (up to 11 characters) used to assign the bottom sheet number, which must be written exactly as in the `.dbf` file (case sensitive).
4. **top_sheet_attribute** Field name (up to 11 characters) used to assign the top sheet number, which must be written exactly as in the `.dbf` file (case sensitive).

This command combines node selection and node set creation from a single shapefile. Its behavior depends on the shape type of the shapefile:

- For Point, PointM, or PointZ ($z$-coordinate ignored) shape types, a node set is created for each point in the shapefile. Within each sheet between the bottom and top sheets (inclusive), the node that is nearest to the point is chosen.
- For PolyLine, PolyLineM, or PolyLineZ ($z$-coordinate ignored) shape types, a node set is created for each polyline in the shapefile. Within each sheet between the bottom and top sheets (inclusive), nodes that fall on or close to the polyline are chosen.
- For Polygon, PolygonM, or PolygonZ ($z$-coordinate ignored) shape types, a node set is created for each polygon in the shapefile. *Each polygon may contain holes and consist of multiple parts but must not be self-intersecting.* Within each sheet between the bottom and top sheets (inclusive), nodes that fall within the polygon are chosen.

Each node set is written to the file named *prefix*`o.node_set.`*set_name*, where *set_name* is read from the **name_attribute** field in the `.dbf` file.

●●●

## Create node set intersection

1. **intersect_set_name** Name of the intersected node set, up to 40 characters.
2. **set_name_1** Name of the first node set, up to 40 characters.

3. **set_name_2** Name of the second node set, up to 40 characters.

Creates a node set with the given name from the shared nodes in **set_name_1** and **set_name_2** node sets. The node set is written to the ASCII output file *prefix*`o.node_set.`-**intersect_set_name**.

●●●

## Create node set union

1. **union_set_name** Name of the union node set, up to 40 characters.
2. **set_name_1** Name of the first node set, up to 40 characters.
3. **set_name_2** Name of the second node set, up to 40 characters.

Creates a node set with the given name from the combined nodes in **set_name_1** and **set_name_2** node sets. The node set is written to the ASCII output file *prefix*`o.node_set.`-**union_set_name**.

●●●

## Create node set difference

1. **diff_set_name** Name of the difference node set, up to 40 characters.
2. **set_name_1** Name of the first node set, up to 40 characters.
3. **set_name_2** Name of the second node set, up to 40 characters.

Creates a node set with the given name from the nodes in **set_name_1** node set that do not belong to **set_name_2** node set. The node set is written to the ASCII output file *prefix*`o.node_set.`**diff_set_name**.

●●●

## Create node set symmetric difference

1. **symdiff_set_name** Name of the symmetric difference node set, up to 40 characters.
2. **set_name_1** Name of the first node set, up to 40 characters.
3. **set_name_2** Name of the second node set, up to 40 characters.

Creates a node set with the given name from the nodes that belong to **set_name_1** or **set_name_2** node sets but not both. The node set is written to the ASCII output file *prefix*`o.node_set.`**symdiff_set_name**.

●●●

---

## Create face set

1. **set_name** Name of the face set, up to 40 characters.

Creates a face set with the given name from the currently selected nodes. The face set is written to the ASCII output file *prefix*`o.face_set.`**set_name**.

●●●

---

## Create segment set

1. **set_name** Name of the segment set, up to 40 characters.

Creates a segment set with the given name from the currently selected nodes. The segment set is written to the ASCII output file *prefix*`o.seg_set.`**set_name**.

●●●

If you have already selected faces, then you can use the command:

---

## Create face set from chosen faces

1. **set_name** Name of the face set, up to 40 characters.

Creates a face set with the given name from the currently selected faces. The face set is written to the ASCII output file *prefix*`o.face_set.`**set_name**.

●●●

If you have already selected segments, then you can use the command:

---

## Create segment set from chosen segments

1. **set_name** Name of the segment set, up to 40 characters.

Creates a segment set with the given name from the currently selected segments. The segment set is written to the ASCII output file *prefix*`o.seg_set.`**set_name**.

●●●

**Important:** When defining a node, face, or segment set, users should take care that the selected nodes, faces, or segments conform to the requirements of a given boundary condition.

For example, the Rain boundary condition expects faces on the surface of the model domain. **HydroGeoSphere** currently does not check that this requirement is met except in a few specific cases. Moreover, users should always select nodes, faces, or segments that belong to the domain for which the boundary condition is defined. **HydroGeoSphere** will assist users by ensuring that each boundary condition operates on only those nodes, faces, or segments that belong to its specified domain. Any nodes, faces, or segments that fall outside the specified domain will be ignored.

### 2.7.3 Type

To define what type of boundary condition (e.g. head, flux etc.) is to be applied use:

---

Type

1. **bc_type** The type of boundary condition to be applied.

The allowable flow boundary condition types are described below in detail.

●●●

#### 2.7.3.1 Specified Head

This is also known as a first-type, Dirichlet, or constant head boundary condition. It is a nodal property so you should first define the subset of nodes for which you want to apply the condition and write them to a nodal data set.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign various specified head boundary conditions:

---

Head

Sets the input type to be a general specified head boundary condition. Note that when applying a specified head boundary condition to the overland flow domain it is treated as "water depth". If the value of the head boundary condition at an overland flow node is exactly equal to the elevation at that node, then this node will behave a like a sink and will not add water to the system, even if the water table is below ground surface.

●●●

For example:

```
boundary condition
    type
    head
```

```
        node set
        inflow

        time value table
        0.0    100.0
        end
    end
```

would define a specified head of 100.0 from time zero for the duration of the simulation for all of the nodes contained in the node set `inflow`.

These heads can be interpolated or turned on and off as discussed in Section 2.7.

## Head equals elevation

Sets the input type to be a special form of the specified head boundary condition where head is set equal to the elevation of the node.

●●●

For example:

```
    boundary condition
        type
        head equals elevation

        node set
        inflow
    end
```

This example shows that a time-value table is not required, since heads are derived from the nodal elevation. However, if you wish to turn the boundary condition on or off, then this can be accomplished by including a Time value table instruction:

```
    boundary condition
        type
        head equals elevation

        node set
        inflow

        time value table
        0.0    1.0
        10.0   -99999.
        end
    end
```

The value 1.0 at time zero is ignored, but the NODATA value −99999 at time 10.0 causes the boundary condition to be turned off and the nodes become unconstrained.

The following command may be used in conjunction with the Head equals elevation boundary condition to offset the prescribed head value.

## Elevation offset

1. **offset** Vertical offset [L] that is applied to the nodal elevation.

The specified head value based on nodal elevation can be increased or decreased by a user specified value by entering a positive or negative offset.

● ● ●

## Head equals initial

Sets the input type to be a special form of the specified head boundary condition where head is set equal to the initial head at the node.

● ● ●

For example:

```
    boundary condition
        type
        head equals initial

        node set
        inflow
    end
```

This condition can be turned on and off as discussed above for Head equals elevation.

Each time you issue a Boundary condition...End instruction, a new boundary condition is formed with a unique ID number. **HydroGeoSphere** processes the boundary conditions in order and later ones may take precedence over later ones. The position in the **grok** file determines the order.

If the node was assigned a specified head or fluid flux value by a previous instruction then it might be overwritten by later head boundary conditions, depending on whether or not both nodes are active at a given time.

#### 2.7.3.2 Specified Flux

This is also known as a second-type, Neumann, specified or constant flux boundary condition. It is an areal or nodal property and so you should first choose the subset of faces/nodes (depending on the BC type) for which you want to apply the boundary condition. These faces/nodes are often (but not always) part of the outer boundary of the grid.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign various specified flux boundary conditions:

---

## Flux

Sets the input type to be a general specified flux boundary condition. The input flux [L $T^{-1}$] is converted internally to a nodal volumetric flow rate [$L^3$ $T^{-1}$] by multiplying by the contributing area of a chosen face.

For example, the following boundary condition declaration would define a specified flux that is mapped from the raster file `recharge.asc` and is applied from time zero for the duration of the simulation for all of the faces contained in the face set `top`.

```
boundary condition
    type
    flux

    face set
    top

    time raster table
    0.0    recharge.asc
    end
end
```

●●●

---

## Rain

Sets the input type to be a general specified flux boundary condition. The input flux [L $T^{-1}$] is converted internally to a nodal volumetric flow rate [$L^3$ $T^{-1}$] by multiplying by the contributing area of a chosen face when projected onto the $xy$-plane.

●●●

---

## Flux nodal

Sets the input type to be a general specified flux boundary condition. The input flux is treated as a nodal volumetric flow rate [$L^3$ $T^{-1}$] that is applied directly to the nodes in the given node set.

For example, the following boundary condition declaration would assign a specified nodal flux that is read from the input file `nflux.txt` and is applied from time zero for the duration of the simulation for all of the nodes contained in the node set `top`. Note that the number of entries in the file must be the same as the number of nodes in the set.

```
boundary condition
    type
    flux nodal

    node set
    top

    time file table
    0.0    nflux.txt
    end

    nodal flux reduction by pressure head
    0.01
    0.1
end
```

•••

These fluxes can be interpolated (see command Interpolate) or turned on and off (see command Nodata value). In cases where flux boundary inputs overlap, fluid fluxes will be accumulated. Note that the definition of wells (Section 2.8.2.4) or tile drains (Section 2.8.2.5) in the model may create a non-zero specified flux boundary condition.

The following instructions can be used to constrain the Flux nodal boundary condition.

## Nodal flux reduction by pressure head

1. **min_pressure_head** Minimum pressure head threshold $\psi_{min}$ [L].
2. **max_pressure_head** Maximum pressure head threshold $\psi_{max}$ [L].

The specified nodal flux can be optionally reduced as a function of the nodal pressure head to avoid pumping from dry materials. Let $Q_0$ denote the specified nodal flux value and $\psi$ the pressure head. Then, for $Q_0 < 0$ the updated nodal flux value $Q$ is defined as

$$Q = \begin{cases} 0, & \psi \leq \psi_{min} \\ \alpha^{2(1-\alpha)} Q_0, & \psi_{min} < \psi < \psi_{max} \\ Q_0, & \psi \geq \psi_{max} \end{cases},$$

where $\alpha = (\psi - \psi_{min})/(\psi_{max} - \psi_{min})$.

●●●

If you want to force the nodal flux to operate within a specified hydraulic head range, you can do so with the following instruction.

## Nodal flux head constraints

1. **inflow_min, inflow_max** Minimum and maximum constraining head values [L] for positive nodal flux.
2. **outflow_min, outflow_max** Minimum and maximum constraining head values [L] for negative nodal flux.

If the nodal flux is positive, and the head at the node is less than **inflow_min**, then the flux will be applied until such time as the head at the node exceeds **inflow_max**. Once the head at the node exceeds **inflow_max**, then the flowrate will be set to zero until such time as the head at the node drops below **inflow_min**.

If the flowrate is negative, and the head at the node is greater than **outflow_min**, then the flowrate will be applied until such time as the head at the node drops below **outflow_max**. Once the head at the node drops below **outflow_max**, then the flowrate will be set to zero until such time as the head at the node exceeds **outflow_min**.

●●●

## Nodal flux head target control

1. **obs_pt_name** Name of the observation point at which total head is evaluated.
2. **domain_name** Name of domain at which total head is evaluated.
3. **interpolate** Logical value (T/F), which if true, causes target head to be interpolated from the time-value table. Otherwise, the value at the left endpoint of a time panel is used.
4. **time(i), head(i)...end** Time [T] and total head target [L] table.

Application of the specified nodal flux can be controlled by the head value at an observation point relative to a target head value. Let $Q_0$ denote the specified nodal flux value, $h$ the head at the observation point, and $h_T$ the target head value. Then the applied nodal flux $Q$ is defined as

$$Q = \begin{cases} Q_0, & Q_0 < 0 \text{ and } h > h_T \\ Q_0, & Q_0 > 0 \text{ and } h < h_T \\ 0, & \text{otherwise.} \end{cases}$$

•••

## Flux nodal by pressure

Sets the input type to be a general specified flux boundary condition. The input flux is treated as a nodal volumetric flow rate [$L^3$ $T^{-1}$] that is applied directly to the nodes in the given node set. The nodal flux is computed from pressure head either through an empirically determined functional stage-discharge relationship (see Pressure-discharge power rating) or through linear interpolation of a stage-discharge table (see Pressure-discharge table). The nodal flux may also be computed from total head via linear interpolation of a stage-discharge table (see Elevation-discharge table).

For example, the following boundary condition declaration would assign a specified nodal flux that is computed from a pressure-discharge table via linear interpolation and is applied to all nodes in the node set `StageDischargeOutflow`. Moreover, the optional Time on/off table command dictates that the boundary condition is active for all times in the range $[12960000, 23328000)$ and is inactive otherwise.

```
boundary condition
    type
    flux nodal by pressure

    name
    StageDischargeOutflow

    node set
    StageDischargeOutflow

    time on/off table
    12960000  23328000
    end

    pressure-discharge table
    0.01   0.0
    0.05  -0.01
    0.1   -0.1
    0.5   -1.0
    end
end
```

•••

## Pressure-discharge power rating

1. **coeff** Power rating curve coefficient [$L^{3-\alpha}$ $T^{-1}$] $C$, where $\alpha$ is the power rating curve exponent defined below.
2. **min_depth** Minimum depth [L] for water flow $\psi_0$.
3. **expon** Power rating curve exponent [-] $\alpha$.

This command defines a functional stage-discharge relationship that is used by the boundary condition Flux nodal by pressure. The specified volumetric flow rate $Q$ [$L^3$ $T^{-1}$] is computed via the equation

$$Q = \begin{cases} C \cdot (\psi - \psi_0)^{\alpha}, & \psi > \psi_0 \\ 0, & \text{otherwise} \end{cases}$$

where $\psi$ is the pressure head.

• • •

## Pressure-discharge table

1. **depth(i), flux(i)...end** Depth [L] and specified volumetric flow rate [$L^3$ $T^{-1}$] table.

This command defines a tabular stage-discharge relationship that is used by the boundary condition Flux nodal by pressure. The specified volumetric flow rate is computed from the table via linear interpolation of pressure head. Note that for any pressure head value that falls outside the table, the flux at the endpoint nearest to that pressure head value will be used.

• • •

## Elevation-discharge table

1. **elevation(i), flux(i)...end** Elevation [L] and specified volumetric flow rate [$L^3$ $T^{-1}$] table.

This command defines a tabular stage-discharge relationship that is used by the boundary condition Flux nodal by pressure. The specified volumetric flow rate is computed from the table via linear interpolation of total head. Note that for any total head value that falls outside the table, the flux at the endpoint nearest to that total head value will be used.

• • •

## Flux nodal from outlet

Sets the input type to be a general specified flux boundary condition that takes water removed by an existing boundary condition (outlet) and injects it at a set of user prescribed nodes (inlet). If $Q$ is the net flux of water removed from the outlet ($Q < 0$), then a flux of

$-Q/n$ is applied at each inlet node, where $n$ is the number of inlet nodes. The following boundary condition types may be applied at the outlet: Flux nodal, Flux nodal by pressure, and Simple drain.

For example, the following boundary condition declaration would assign a Simple drain boundary condition at the outlet and would then link that boundary condition to a set of surface nodes.

```
! Setup outlet boundary condition
use domain type
surface

clear chosen nodes

choose node
25 0 8

create node set
nsimpledrain

boundary condition
    type
    simple drain

    name
    nsimpledrain

    node set
    nsimpledrain

    time value table
    0 0.1 1e8
    end
end

! Re-injection boundary condition
clear chosen nodes
choose node
10 0 10

create node set
surface_dump

boundary condition
    type
    flux nodal from outlet
```

```
        name
        surface_dump

        node set
        surface_dump

        outlet bc name
        nsimpledrain
    end
```

•••

## Outlet bc name

1. **bc_name** Name of the outlet boundary condition, at most 40 characters.

Sets the name of the outlet boundary condition for the Flux nodal from outlet boundary condition. The input name must match the name of an existing specified flux boundary condition.

•••

## Irrigation on demand

Sets the input type to be a general specified flux boundary condition that adds a prescribed volume of water to the surface of the model over a given time duration. An irrigation event is triggered if the pressure head at a user specified observation point in the porous media domain drops below a user specified threshold. The volume of water added is computed by multiplying the contributing area of a chosen face when projected onto the $xy$-plane with a user specified water height applied uniformly over the set of chosen faces. An irrigation event can occur only during the current growing season, which is defined by a time on/off table. Moreover, within a growing season, irrigation events are restricted to occur at fixed time intervals defined by the irrigation recurrence interval. For example:

```
    boundary condition
        type
        irrigation on demand

        face set
        top

        irrigation parameters
        observation_pt     ! name of observation point
        -10                ! pressure head threshold
        0.015              ! total height of water applied
        86400              ! duration of irrigation event
        604800             ! recurrence interval of irrigation
        12960000 23328000  ! time on and time off in year 1
        44496000 54864000  ! time on and time off in year 2
        end

        pressure irrigation table
         -1 0.015
         -2 0.035
         -3 0.045
         -4 0.05
        -10 0.075
        end
    end
```

In this example, irrigation events of duration one day (86400 s) can occur once per week (604800 s) within each growing season. The first potential irrigation event is at the start of the first growing season (12960000 s). The input parameters are specified by the command Irrigation parameters and optionally by the command Pressure irrigation table described below.

• • •

## Irrigation parameters

1. **obs_pt_name** Name of the observation point at which pressure head is evaluated to trigger an irrigation event.
2. **thresh** Pressure head [L] threshold to trigger an irrigation event.
3. **height** Height [L] of water applied during an irrigation event.
4. **duration** Time duration [T] of an irrigation event.
5. **interval** Irrigation recurrence interval [T]. It is required that **duration** < **interval**.
6. **ton(i), toff(i)...end** Time on [T] and time off [T] table. The time values should satisfy the following set of inequalities:

$$t_{\text{on}}(1) < t_{\text{off}}(1) < t_{\text{on}}(2) < t_{\text{off}}(2) < \cdots < t_{\text{on}}(n) < t_{\text{off}}(n),$$

   where $n$ is the number of panels. An irrigation event can occur at time $t$ if and only if $t_{\text{on}}(i) \leq t < t_{\text{off}}(i)$ for some $i \in \{1, \ldots, n\}$.

This command specifies the input parameters for the boundary condition type Irrigation on demand.

●●●

## Pressure irrigation table

1. **pressure(i), height(i)...end** Pressure head [L] and irrigation water height [L] table.

This command specifies that the height of water applied for the duration of an irrigation event be dependent on the pressure head at the observation point. It is an optional command, which if used, causes the water height to be defined via linear interpolation from the given table. In the absence of this command, the height of water applied during an irrigation event is given by the value specified in the command Irrigation parameters.

●●●

#### 2.7.3.3 Fluid Transfer

The fluid transfer boundary condition is a third-type, or Cauchy boundary condition. It takes as input a user prescribed reference head value at a distance $\ell$ [L]. The direction of fluid flow, either in, or out, of the boundary condition, is determined by the difference between total head and the reference head at a distance. The volumetric flow rate is defined as the product of the head difference and a conductance $C$ [L$^2$ T$^{-1}$] defined from a user prescribed hydraulic conductivity $K$ [L T$^{-1}$] that represents an intermediate material via the equation

$$C = \frac{K \cdot A}{\ell},$$

where $A$ is the area of a element face. This boundary condition should be assigned to the faces at the edge (side) of a model to represent the regional influence on the model domain.

## Fluid transfer

Sets the input type to be a fluid transfer boundary condition. For example:

```
boundary condition
    type
    fluid transfer

    face set
    outflow

    time value table
    ! time  ref head
```

```
              0     100
            end

            fluid transfer coefficients
            1e-5  ! hydraulic conductivity
            1000  ! distance
        end
```

would assign a fluid transfer with a head value of 100 m at a distance of 1000 m and a hydraulic conductivity of $10^{-5}$ m/s for the material external to the boundary condition.

•••

The fluid transfer boundary condition requires the following command to define its input parameters.

---

## Fluid transfer coefficients

1. **cond** Hydraulic conductivity [L $T^{-1}$].
2. **dist** Distance [L].

Defines the hydraulic conductivity and distance parameters for the fluid transfer boundary condition.

•••

#### 2.7.3.4 Free Drainage

Assigns a free drainage boundary condition, as described in Section 3.8.1 of the Theory Manual to nodes on all faces in the specified set. These faces should be on the surface of the domain.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign free drainage boundary conditions:

---

## Free drainage

Sets the input type to be a free drainage boundary condition.

•••

For example:

```
    boundary condition
        type
        free drainage

        face set
        outflow
    end
```

This example shows that a time-value table is not mandatory. However, if you wish to turn the free drainage boundary condition on or off, then this can be accomplished by including a Time value table instruction:

```
    boundary condition
        type
        free drainage

        face set
        outflow

        time value table
        0.0         1.0
        10.0   -99999.0
        end
    end
```

The value 1.0 at time zero is ignored, but the NODATA value −99999 at time 10.0 causes the boundary condition to be turned off and the nodes become unconstrained.

#### 2.7.3.5 Potential Evapotranspiration

Potential evapotranspiration is a specified flux [L $T^{-1}$] applied as a second-type boundary condition and is used in conjunction with evapotranspiration properties, which can be defined as described in Section 2.8.5. It is an areal property and so you should first choose the subset of faces for which you want to apply the condition. These faces should be part of the top boundary of the grid and, in this case, the top boundary must be coincident with the 2-D slice that was used to define the 3-D mesh. This restriction arises because of grid numbering assumptions that are made when applying evapotranspiration as a function of depth. Note also that if ET domains are not defined and this boundary condition is applied, then **grok** will stop and issue a warning message.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign potential evapotranspiration:

## Potential evapotranspiration

Sets the input type to be a potential evapotranspiration boundary condition. The input flux [L $T^{-1}$] is converted internally to a nodal volumetric flow rate [$L^3$ $T^{-1}$] by multiplying by the contributing area of a chosen face. This boundary condition requires a surface flow domain using the dual-node approach (see Dual nodes for surface flow). It can be applied to either the surface flow or porous media domains. Actual evapotranspiration is calculated from potential evapotranspiration by accounting for surface evaporation in the surface flow domain and subsurface evaporation and transpiration in the porous media domain. For example:

```
boundary condition
    type
    potential evapotranspiration

    face set
    top

    time value table
    0.0    0.001
    end
end
```

would define a potential evapotranspiration flux of 0.001 (with units [L $T^{-1}$]) from time zero for the duration of the simulation for all nodes contained in the face set `top`.

•••

#### 2.7.3.6 River Flux

Assigns a river flux boundary condition, as described in Section 3.8.1 to nodes in the specified set. These nodes are normally located on the surface of the domain. For river flux nodes, water may flow in or out of the domain depending on the difference in head between the river node and the specified river head value.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a river flux boundary condition:

---

### Simple river

Sets the input type to be a river flux drainage boundary condition.

•••

For example:

```
boundary condition
    type
    simple river
```

```
        node set
        my_river

        time value table
        0.0      25.     1.e-5
        end
    end
```

This example shows the use of a Time value table instruction to define the river head (25) and river conductance ($10^{-5}$) values. This would not be very useful in most cases, since conductance and head vary from node to node along a river. In such cases, the inputs would be defined by a Time file table, with unique values given for each node:

```
    boundary condition
        type
        simple river

        node set
        my_river

        time file table
           0.0    my_river.lst
        1000.     none
        end
    end
```

Here, the river flux boundary condition would become inactive after time 1000 and the nodes would become unconstrained. In this case, the file `my_river.lst` would contain two values per line, the river conductance and head value, and one line for each node in the set.

#### 2.7.3.7 Drain Flux

Assigns a drain flux boundary condition, as described in Section 3.8.1 of the Theory Manual to nodes in the specified set. These nodes are normally located on the surface of the domain. The fundamental difference between river and drain flux nodes is that drain nodes only allow water to flow out of the system, depending on the difference in head between the drain node and the specified pressure head value.

The following instruction can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a drain flux boundary condition:

---

Simple drain

Sets the input type to be a drain flux drainage boundary condition.

•••

For example:

```
boundary condition
    type
    simple drain

    node set
    my_drain

    time value table
    0.0      25.      1.e-5
    end
end
```

This example shows the use of a Time value table instruction to define the pressure head (25) and drain conductance $10^{-5}$ values. This would not be very useful in most cases, since conductance and head would vary from drain node to drain node. In such cases, the inputs would be defined by a Time file table, with unique values given for each node:

```
boundary condition
    type
    simple drain

    node set
    my_drain

    time file table
       0.0    my_drain.lst
    1000.     none
    end
end
```

Here, the drain flux boundary condition would become inactive after time 1000 and the nodes would become unconstrained. In this case, the file `my_drain.lst` would contain two values per line, the drain conductance and head value, and one line for each node in the set.

#### 2.7.3.8 Tunnels

Assigns a tunnel boundary condition, as described in Section 3.8.1 of the Theory Manual to segments in the specified set. These segments should belong to the porous medium domain. The fundamental difference between a Tunnel and Simple drain boundary condition is that volumetric fluxes into the tunnel are calculated at the surface of an imaginary cylinder whose dimensions are defined by the user (i.e., the tunnel radius) and the model mesh (i.e., the length of tunnel segments). Furthermore, the tunnel may or may not be surrounded by zones with distinct hydrogeologic properties (i.e., the excavation damaged zone and grouted zone).

**Important:** this boundary condition is defined only for block hexahedral meshes; triangular prism meshes are not supported.

The following instruction can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a Tunnel boundary condition:

## Tunnel

Sets the input type to be a tunnel boundary condition.

●●●

For example:

```
    boundary condition
        type
        tunnel

        name
        TBC

        segment set
        sTunnel

        time value table
        0.00  0.00  0.01
        end
    end
```

This example shows the use of a Time value table instruction to define the pressure head (0.00) and tunnel radius (0.01) values. This command allows you to define time-varying properties for the segment set `sTunnel`, which is useful in cases where the tunnel is well established and its location or footprint is not changing over time. In cases where the tunnel geometry changes over time (e.g., as a result of active digging) the inputs can be defined by a Time file table, with unique values given for each segment. For example:

```
    boundary condition
        type
        tunnel

        name
        TBC

        segment set
        sTunnel

        time file table
        0.0    Tunnel_properties_day0.lst
        86400  Tunnel_properties_day1.lst
        end
    end
```

Here, the tunnel boundary condition would be updated after 1 day (86400 seconds). In this case, the file `Tunnel_properties_day0.lst` and `Tunnel_properties_day1.lst` would contain two values per line, the pressure head and tunnel radius values, and one line for each segment in the set. In this way you can define both spatially and temporally varying tunnel properties.

Fluxes into the tunnel may or may not be impacted by the excavation damaged zone (EDZ) and grouted zone (GZ). These are distinct zones with distinct hydrogeologic properties that separate the open tunnel from the surrounding porous medium. The following two optional instructions can be used to parameterize the EDZ and GZ. Note that if these instructions are not used, then the EDZ and GZ are assumed to have a thickness of zero and their conductivity is set to that of the porous medium.

## Tunnel edz thickness and conductivity

1. **edz_case** Type of EDZ conductivity (1 or 2) corresponding to input **edz_cond**. If 1 is entered, then **edz_cond** is given directly as the conductivity of the EDZ. Otherwise, if 2 is entered, then **edz_cond** is the ratio of EDZ conductivity to the porous medium hydraulic conductivity.

2. **edz_thickness, edz_cond** Thickness [L] and either the conductivity [L $T^{-1}$] or conductivity ratio [-] depending on the value of input **edz_case**.

Sets the EDZ thickness and conductivity for the tunnel boundary condition. Note that if **edz_case** = 2, then the EDZ conductivity is defined as

$$K_{\text{EDZ}} = \textbf{edz\_cond} \cdot K,$$

where $K$ is the hydraulic conductivity of the porous medium.

●●●

## Tunnel grout thickness and conductivity

1. **gz_case** Type of GZ conductivity (1 or 2) corresponding to input **gz_cond**. If 1 is entered, then **gz_cond** is given directly as the conductivity of the GZ. Otherwise, if

2 is entered, then **gz_cond** is the ratio of GZ conductivity to the porous medium hydraulic conductivity.

2. **gz_thickness, gz_cond_thresh, gz_cond** Thickness [L], grout conductivity threshold [L $T^{-1}$], and either the conductivity [L $T^{-1}$] or conductivity ratio [-] depending on the value of input **gz_case**.

Sets the GZ thickness, conductivity ratio, and conductivity for the tunnel bounary condition. The GZ conductivity is defined as

$$K_{\mathrm{GZ}} = \begin{cases} K, & K < \textbf{gz\_cond\_thresh} \\ K'_{\mathrm{GZ}}, & K \geq \textbf{gz\_cond\_thresh} \end{cases}$$

where $K$ is the hydraulic conductivity of the porous medium and

$$K'_{\mathrm{GZ}} = \begin{cases} \textbf{gz\_cond}, & \textbf{gz\_case} = 1 \\ \textbf{gz\_cond} \cdot K, & \textbf{gz\_case} = 2 \end{cases}$$

●●●

For example, the boundary condition declaration would create a tunnel with defined pressure head (0.00) and tunnel radius (0.01), assigned to the segment set `sTunnel`. The tunnel includes an excavation damage zone with a defined thickness (0.25) and conductivity expressed as a ratio (1.5) of the porous medium conductivity. The tunnel also includes a grout zone with a defined thickness (0.1) and conductivity expressed as a ratio (0.1) of the porous medium conductivity, which is not allowed to exceed the maximum grout conductivity (3.5).

```
boundary condition
    type
    tunnel

    name
    TBC

    segment set
    sTunnel

    time value table
    0.00  0.00  0.01
    end

    tunnel edz thickness and conductivity
    2
    0.25  1.5
```

```
        tunnel grout thickness and conductivity
        2
        0.1  3.5  0.1
    end
```

### 2.7.3.9 Makeup Water

The boundary condition Makeup water is similar to Simple drain in that its behavior is controlled by the difference between the pressure head at a specified set of nodes and the specified head value. These nodes typically belong to the surface domain. The fundamental difference between them is that Makeup water only allows water to flow into the system, whereas Simple drain only allows water to flow out of the system. The following instruction can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a Makeup water boundary condition:

---

#### Makeup water

Assigns a makeup water boundary condition to nodes in the specified set. These nodes can be located in either the surface or porous media domains. The makeup water boundary condition adds water to the system if the pressure head falls below the specified value and becomes unconstrained if head is greater than the specified value. Note that water is only allowed to enter the system and cannot exit the system. For example:

```
clear chosen nodes
choose node
100 100 100

create node set
nmakeup

boundary condition
    type
    makeup water

    node set
    nmakeup

    time value table
    0.0 1.25 1e8
    end
end
```

This example shows the use of a Time value table instruction to define the pressure head (1.25) and conductance ($10^8$) values. In this case, the boundary condition will prevent pressure head from dropping below 1.25 m. The high conductance value ensures that there is no resistance to the addition of water.

●●●

#### 2.7.3.10 Surface Flow

Critical depth and zero-depth gradient boundary conditions can be assigned to segments within a specified set as described in Section 3.8.2 of the Theory Manual. These segments should be part of the overland flow domain and are typically located on the outer boundary. The reservoir and reservoir with spillway boundary conditions can be assigned to a node set consisting of a single node that belongs to either the overland flow domain or the 1-D channel flow domain.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group.

### Critical depth

Sets the input type to be a critical depth boundary condition. For example:

```
boundary condition
    type
    critical depth

    segment set
    outflow
end
```

This example shows that a time-value table is not mandatory. However, if you wish to turn the critical depth boundary condition on or off, then this can be accomplished by including a Time value table instruction:

```
boundary condition
    type
    critical depth

    segment set
    outflow

    time value table
     0.0        1.0
    10.0   -99999.0
    end
end
```

The value 1.0 at time zero is ignored, but the NODATA value −99999 at time 10.0 causes the boundary condition to be turned off and the nodes become unconstrained.

●●●

## Zero depth gradient

Sets the input type to be a zero-depth gradient boundary condition. For example:

```
boundary condition
    type
    zero depth gradient

    segment set
    outflow

    bed slope
    0.1
end
```

The zero-depth gradient boundary condition can be turned on or off by including a Time value table instruction.

●●●

## Bed slope

1. **slope** Bed slope [-].

This command specifies the bed slope for the zero-depth gradient boundary condition, $S_0$ in Equation 3.57.

●●●

The reservoir and reservoir with spillway boundary conditions facilitate the simulation of surface water management schedules that aim to remove water from instream flow (overland flow and/or 1-D channel flow domains), store water, and release water into stream flow. While water removal from instream flow is dependent upon water being present in the requisite domain, the storage is managed as an offline numerical reservoir, hence volumetric storage is independent of mesh discretization and topographic resolution.

## Reservoir

Sets the input type to be the reservoir boundary condition. This boundary condition may be active in either the overland flow or 1-D channel flow domains. For example:

```
boundary condition
    type
    reservoir

    name
    north_dam

    node set
    north_dam_control_node

    time value table
          0  0.5
     2592000  0
     7776000 -0.5
    10368000  0
    end
    !----------------------------reservoir storage details
    initial reservoir storage
    0
    maximum reservoir storage
    1500000
    base reservoir storage
    100000
    !----------------------------instream water withdrawl control
    nodal flux reduction by pressure head
    0.05
    0.25

    tecplot output
end
```

In the above example, water would be removed from the target node in the overland (or channel) domain at a rate of 0.5 m$^3$ s$^{-1}$ between 0 s and 2592000 s, and then released from storage back into the target node at a rate of 0.5 m$^3$ s$^{-1}$ between 7776000 s and 10368000 s. The pressure head endpoints specify the pressure (water depth) range wherein withdrawal rates are reduced (via linear interpolation) on account of limited water availability, with withdrawal stopped below a pressure threshold of 0.05 m, and with the full withdrawal rate realized at pressures above 0.25 m.

● ● ●

The reservoir boundary condition writes timeseries information to the Tecplot ASCII output file named *prefix*`o.reservoir_`*bcname*`.dat`, where *bcname* is the name assigned to the boundary condition. This file reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Pressure Head` | [L] | Pressure head at reservoir node. |
| `Rate of Change` | [$L^3$ $T^{-1}$] | Rate of storage change. |
| `Storage` | [$L^3$] | Reservoir storage. |
| `Scheduled Rate` | [$L^3$ $T^{-1}$] | User specified rate of storage change. |
| `Scheduled Storage` | [$L^3$] | Reservoir storage computed from scheduled rate of storage change. |

The following boundary condition is a variation of the reservoir boundary condition described above.

## Reservoir with spillway

Sets the input type to be the reservoir with spillway boundary condition. This boundary condition may be active in either the overland flow or 1-D channel flow domains. Although similar to the reservoir boundary condition, there are some important differences:

- Inflow to the reservoir is defined via a hydrograph (see Inflow hydrograph name).
- Discharge from the reservoir consists of 1) spillway discharge from surcharge storage given by a spillway discharge curve (see Spillway parameters), 2) gate discharge from maximum storage given by a time-value table (see Gate discharge table), and 3) overflow discharge if the amount of inflowing water exceeds the storage capacity of the reservoir.

See Figure 2.9 for a diagram of our reservoir with spillway conceptualization. Timeseries information that details the state of the reservoir is written to the Tecplot ASCII output file *prefix*`o.reservoir_spillway.`*bcname*`.dat`, where *bcname* is the name assigned to the boundary condition. For example:

```
boundary condition
    type
    reservoir with spillway

    name
    north_dam

    node set
    north_dam_control_node

    inflow hydrograph name
    reservoir_inflow

    !-----------------------------reservoir storage details
    initial reservoir storage
```

```
      0.0
      base reservoir storage
      400.0
      maximum reservoir storage
      2000.0
      surcharge reservoir storage
      2000.0

      !------------------------------reservoir discharge details
      spillway parameters
      2.0 ! spillway constant
      1.0 ! spillway length
      0.0     1.0 ! volume-elevation table
      500.0   1.5
      1000.0  1.9
      1500.0  2.2
      2000.0  2.4
      end

      gate discharge table
      0.0    0.0
      600.0  0.1
      1200.0 0.1
      1800.0 0.3
      2400.0 0.5
      3000.0 0.6
      3600.0 0.6
      end
  end
```

We note that this boundary condition supports the Nodal flux reduction by pressure head command, which may be used to limit inflow to the reservoir. We recommend this option in practice, to avoid removing too much water from the system, which can lead to model instability.

●●●

The reservoir with spillway boundary condition writes timeseries information to the Tecplot ASCII output file named *prefix*`o.reservoir_spillway.`*bcname*`.dat`, where *bcname* is the name assigned to the boundary condition. This file reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Olf Pressure Head` | [L] | Overland flow pressure head at reservoir node. |

| | | |
|---|---|---|
| `Channel Pressure Head` | [L] | Channel flow pressure head at reservoir node. |
| `Initial Storage` | [$L^3$] | Reservoir storage at start of timestep. |
| `Storage` | [$L^3$] | Reservoir storage at simulation time. |
| `Rate of Storage Change` | [$L^3$ $T^{-1}$] | Rate of storage change. |
| `Elevation` | [L] | Reservoir water elevation. |
| `Rate of Elevation Change` | [L $T^{-1}$] | Rate of water elevation change. |
| `Inflow Rate` | [$L^3$ $T^{-1}$] | Inflow rate to reservoir from hydrograph. |
| `Storage Loss Rate` | [$L^3$ $T^{-1}$] | Rate of storage loss from reservoir. |
| `Gate Discharge Rate` | [$L^3$ $T^{-1}$] | Rate of discharge from maximum storage. |
| `Spillway Discharge Rate` | [$L^3$ $T^{-1}$] | Rate of discharge from surcharge storage. |
| `Overflow Rate` | [$L^3$ $T^{-1}$] | Rate of water overflowing reservoir. |
| `Total Discharge Rate` | [$L^3$ $T^{-1}$] | Sum of the three prior values. |
| `Error` | [$L^3$ $T^{-1}$] | Mass balance error (Equation 2.8). |
| `Error Rel` | [-] | Relative mass balance error (Equation 2.9). |
| `Error Percent` | [-] | Percent mass balance error (Equation 2.10). |

Note that mass balance errors reported in the table are computed with respect to the reservoir inflow rate, total discharge rate, storage loss rate, and rate of storage change. If the boundary condition is applied to the overland flow domain, then the no data value ($-9999.0$) is reported for the channel pressure head. If a storage to elevation table was not defined, then the no data value ($-9999.0$) is reported for elevation and rate of elevation change.

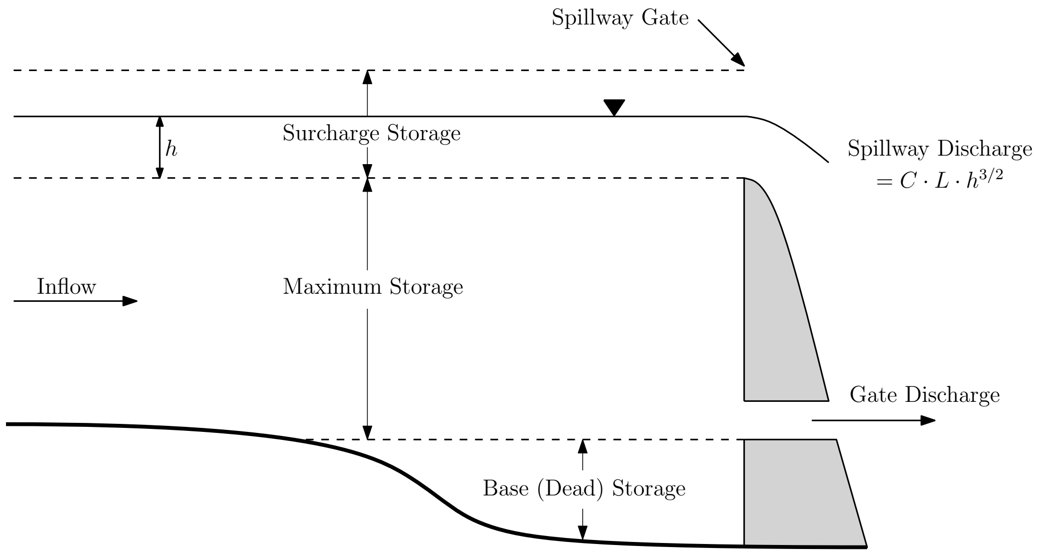


Figure 2.9: Diagram of a reservoir with a spillway. The total storage of the reservoir is the sum of its dead, maximum, and surcharge storage.

The following commands may be used by either the reservoir or reservoir with spillway boundary conditions.

## Initial reservoir storage

1. **init_storage** Reservoir initial storage [L$^3$].

Sets the initial storage of the reservoir.

• • •

## Base reservoir storage

1. **base_storage** Reservoir base (dead) storage [L$^3$].

Sets the base or dead storage of the reservoir (see Figure 2.9).

• • •

## Maximum reservoir storage

1. **max_storage** Reservoir maximum storage [L$^3$].

Sets the maximum storage of the reservoir (see Figure 2.9).

• • •

The following commands may be used by only the reservoir with spillway boundary condition.

## Specified reservoir storage

1. **time(i), storage(i)...end** Time [T] and reservoir storage [L$^3$] list.

For each time $t$ in the input list, reservoir storage is set to the corresponding storage value at the beginning of the **HydroGeoSphere** timestep that starts at time $t$, i.e., $[t, t + \Delta t]$. This command may be used, for example, to match a time series of observed reservoir storage values.

• • •

## Surcharge reservoir storage

1. **surcharge_storage** Reservoir surcharge storage [L$^3$].

Sets the surcharge storage of the reservoir (see Figure 2.9).

• • •

## Spillway parameters

1. **const** Spillway constant $C$ [$L^{1/2}$ $T^{-1}$].
2. **length** Spillway length $L$ [L].
3. **storage(i), elevation(i)...end** Storage [$L^3$] and elevation [L] table.

Sets the parameters that define the spillway discharge rate from the surcharge storage area of the reservoir. The spillway discharge rate is defined by the function

$$\text{spillway discharge rate} = C \cdot L \cdot h^{3/2}$$

where $h$ is the water elevation in surcharge storage, which is computed as follows. Let $f$ denote the piecewise linear function defined by the storage-elevation table, let $S_d$ denote the base storage, let $S_m$ denote the maximum storage, and let $S$ denote the reservoir storage. Then

$$h = f(S) - f(S_d + S_m), \quad S \geq S_d + S_m$$

For storage values that fall outside the table, the nearest elevation is used.

●●●

## Gate discharge table

1. **time(i), discharge_rate(i)...end** Time [T] and reservoir gate discharge rate [$L^3$ $T^{-1}$] table.
2. **interpolate** Logical value (T/F), which if true, causes gate discharge rates to be interpolated from the time-value table. Otherwise, the value at the left endpoint of a time panel is used.

Sets the reservoir gate discharge table that controls discharge from the maximum storage area of the reservoir. For time values that fall outside the table, the nearest discharge rate is used.

●●●

## Inflow hydrograph name

1. **hydrogaph_name** Inflow hydrograph name, up to 80 characters.

Sets the name of the hydrograph that defines the inflow rate to the reservoir. If the reservoir is defined in the channel flow domain, then the inflow rate is the sum of the surface and channel flow rates, otherwise, the inflow rate is the same as the surface flow rate. The hydrograph must be defined via the command Set hydrograph nodes prior to issuing this command. We recommend that the hydrograph node set contains the node at which the reservoir is defined.

•••

The following optional command may be used to define a loss of water from reservoir storage over time. Any water that is lost from reservoir storage via this command is also removed from the model itself.

---

## Reservoir storage loss table

1. **time(i), loss_rate(i)...end** Time [T] and reservoir storage loss rate [$L^3$ $T^{-1}$] table.
2. **interpolate** Logical value (T/F), which if true, causes storage loss rates to be interpolated from the time-value table. Otherwise, the value at the left endpoint of a time panel is used.

Sets the reservoir storage loss table that controls water lost from reservoir storage and the model. Naturally, the amount of water lost is constrained by the current amount of water stored in the reservoir. For time values that fall outside the table, the nearest storage loss rate is used.

•••

#### 2.7.3.11 Snowmelt

You can assign a snowmelt or a rain and snowmelt boundary condition to faces in the specified set. These faces should be part of the surface flow domain.

The following instruction can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a snowmelt boundary condition.

---

## Snowmelt

Sets the input type to be a snowmelt boundary condition. For example:

```
boundary condition
   type
   snowmelt

   face set
   top

   time value table
```

```
      !  time             snowfall            air temperature
           0              2.5848E-07          -7
           2592000        2.0447E-07          -6.1
           5184000        2.5463E-07          -0.9
           7776000        0                    6.2
           10368000       0                    12.9
           12960000       0                    18
           15552000       0                    20.2
           18144000       0                    19.3
           20736000       0                    14.8
           23328000       0                    8.6
           25920000       0                    2.4
           28512000       2.7006E-07          -4
           31104000       2.7006E-07          -4
      end

      snowmelt constants
      100.0             ! snow density
      5.78704E-06       ! melting constant
      0.0               ! sublimation constant
      0.0               ! threshold temperature
      0.1               ! initial snow depth

      interpolate

      tecplot output
   end
```

This example shows the use of a Time value table instruction to define the snowfall and air temperature values. The density of snow, melting constant, rate of sublimation, threshold temperature, and initial snow depth are assigned via the Snowmelt constants instruction.

●●●

The snowmelt boundary condition writes timeseries information to the Tecplot ASCII output file named *prefix*`o.snow_balance.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Depth` | [L] | Snow depth. |
| `Snowfall` | [L $T^{-1}$] | Rate of snowfall. |
| `Air Temperature` | [°C] | Air temperature. |
| `Melting` | [L $T^{-1}$] | Rate of snow melting. |
| `Sublimation` | [L $T^{-1}$] | Rate of snow sublimation. |

The following command defines the input parameters for the snowmelt boundary condition and must be specified as part of its definition.

## Snowmelt constants

1. **snow_rho** Snow density [M $L^{-3}$].
2. **snow_melt** Snow melting constant [M $L^{-2}$ $T^{-1}$ $°C^{-1}$].
3. **snow_sublimation** Snow sublimation constant [M $L^{-2}$ $T^{-1}$].
4. **snow_threshold_temp** Snow threshold temperature [°C].
5. **snow_initial_depth** Initial snow depth [L].

Defines values for the snow density, snow melting constant, snow sublimation constant, snow threshold temperature, and the initial snow depth.

● ● ●

The snow melting constant may also be defined via a time-value table.

## Time varying snow melting constant

1. **time(i), snow_melt(i)...end** Time [T] and snow melting constant [M $L^{-2}$ $T^{-1}$ $°C^{-1}$] table.

Assigns the snow melting constant from the table. At time **time(i)** the snow melting constant **snow_melt(i)** is applied and maintained until time **time(i+1)**. The last snow melting constant entered in the table will be applied until the end of the simulation. At any time before the first time in the table, the snow melting constant specified by the command Snowmelt constants is used.

● ● ●

## Rain and snowmelt

Sets the input type to be equivalent liquid precipitation whereby rainfall and snowfall are partitioned based on air temperature (via the threshold temperature), and snowmelt is calculated using air temperature and the melting constant. The rainfall portion of the total precipitation enters canopy storage (if specified as non-zero), while the snowfall portion is assumed to fall directly on the ground and does not interact with the canopy. The rain and snow partitioning and snowmelt calculations are performed every timestep and therefore have a subsequent contribution to model run time. Precipitation is specified as a liquid water equivalent rate [L $T^{-1}$] in either a time-value or time-raster table along with air temperature [°C].

```
boundary condition
    type
    rain and snowmelt

    name
    rainfall_and_snow

    face set
    top

    time raster table
    0          ./precip/pcp_month1.asc  ./temp/T_month1.asc
    2678400    ./precip/pcp_month2.asc  ./temp/T_month2.asc
    5356800    ./precip/pcp_month3.asc  ./temp/T_month3.asc
    7948800    ./precip/pcp_month4.asc  ./temp/T_month4.asc
    10627200   ./precip/pcp_month5.asc  ./temp/T_month5.asc
    end
end
```

This example shows the use of a Time raster table command to define the precipitation and air temperature values. The density of snow, melting constant, rate of sublimation, threshold temperature, and initial snow depth must be assigned by the commands: Snow density, Melting constant, Sublimation constant, Threshold temperature, and Initial snow depth, respectively, in the surface flow properties file (`.oprops`; see Section 2.8.4). Note that initial snow depth can also be specified for all elements in the surface flow domain from a previously generated output file via the grok command Initial snow depth from output file.

●●●

The rain and snowmelt boundary condition writes time series information to the Tecplot ASCII output file named *prefix*`o.Rain_SnowMelt_balance.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `TotalP` | [$L^3$ $T^{-1}$] | Total liquid precipitation rate over the surface domain. |
| `Liquid` | [$L^3$ $T^{-1}$] | Rainfall rate over the surface domain. |
| `Solid` | [$L^3$ $T^{-1}$] | Snowfall rate as liquid equivalent over the surface domain. |
| `Snowmelt` | [$L^3$ $T^{-1}$] | Snowmelt rate as liquid equivalent over the surface domain. |
| `Sublimation` | [$L^3$ $T^{-1}$] | Sublimation rate as liquid equivalent over the surface domain. |
| `SnowDepthChange` | [$L^3$ $T^{-1}$] | Snow depth change (net of accumulation and melting) as liquid equivalent over the surface domain. |

The rain and snowmelt boundary condition also generates the following binary output files (see Appendix A):

*prefix*`o.liquid_p.0001`

*prefix*`o.solid_p.0001`

*prefix*`o.snowmelt.0001`

*prefix*`o.snowdepth.0001`

*prefix*`o.snowcover.0001`

that are processed by **HSPLOT** or **HGS2VTU** for visualization.

### 2.7.4 Node, Face, and Segment Sets

Some boundary condition types are applied to nodes (e.g., head) while others are applied to faces (e.g., flux), or to segments (e.g., critical depth). To define a group of nodes use:

#### Node set

1. **bc_set_name** Name of the node set, up to 40 characters.

Applies the boundary condition to the specified node set.

●●●

To define a group of faces use:

#### Face set

1. **bc_set_name** Name of the face set, up to 40 characters.

Applies the boundary condition to the specified face set.

●●●

To define a group of segments use:

---

## Segment set

1. **bc_set_name** Name of the segment set, up to 40 characters.

Applies the boundary condition to the specified segment set.

●●●

### 2.7.5 Time-Varying Inputs

To define more complex spatial and temporal properties for a boundary condition the following commands may be used.

---

## Time value table

1. **bc_time(i), bc_val(i)...end** Time [T] and boundary condition value list.

At time **bc_time(i)** the value **bc_val(i)** is applied and maintained until time **bc_time(i+1)**. The last value entered in the list will be applied until the end of the simulation.

●●●

---

## Time raster table

1. **bc_time(i), bc_raster(i), (band(i))...end** Time [T], raster filename, and optional raster band id (default value 1) list.

For the rain and snowmelt, simple drain, makeup water, and simple river boundary conditions the input format becomes:

1. **bc_time(i), bc_raster1(i), bc_raster2(i), (band1(i), band2(i))...end** Time [T], raster1 filename, raster2 filename, and optional raster band ids (default value 1) list.

All filenames with spaces must be enclosed by double quotes (“”). To summarize, the valid input formats are:

- time, filename (band = 1 by default)
- time, filename, band
- time, filename1, filename2 (band1 = band2 = 1 by default)
- time, filename1, filename2, band1 (band2 = 1 by default)
- time, filename1, filename2, band1, band2

At time **bc_time(i)** the data in **band(i)** of raster file **bc_raster(i)** is applied and maintained until time **bc_time(i+1)**. The last raster file entered in the list will be applied until the end of the simulation. **HydroGeoSphere** uses the node (or face centroid) *xy*-coordinate for each member of the node/face set to interpolate a value for the boundary condition at that point.

●●●

The following instructions use the same input data structure except they are applied to *xz*- or *yz*-coordinates:

Time raster xz table
Time raster yz table

## Time file table

1. **bc_time(i), bc_file(i)...end** Time [T] and filename list.

At time **bc_time(i)** the values read from file **bc_file(i)** are applied and maintained until time **bc_time(i+1)**. The data from the last file entered in the list will be applied until the end of the simulation. **HydroGeoSphere** reads a list of values from the file and assigns them in order to the current set of nodes or faces, depending on what type of boundary condition is being applied. The number of values in the file must match the number of nodes or faces in the set or **grok** will stop with an error message. Note that the first value in the file must be the number of data values in the file.

●●●

This instruction can be used in conjunction with other instructions to create files of nodal values which are then read to define boundary conditions. For example, these instructions:

```
clear chosen nodes
choose nodes x plane
0.0
1.e-5

nodal function z to file
```

```
        inflow.txt
        ! Z   value
            0.0   0.
            3.   10.
        10.    4.
    end

    create node set
    inflow
```

create a file `inflow.txt` which contains a value for each currently chosen node. It uses the nodal $z$-coordinate to interpolate the value from the tabulated function of $z$-coordinate versus value. We can now use the file like this:

```
    boundary condition
        type
        head

        node set
        inflow

        time file table
        0.0   inflow.txt
        11.   none
        end
    end
```

Note that we use the node set inflow in conjunction with the file `inflow.txt`. Any mismatch in the number of nodes between the node set file and the data file will cause **grok** to stop with an error message.

Input files for the Time file table command can also be generated for boundary conditions defined in terms of a node selection (via Create node set) by the commands: Make time file table input xyz, Make time file table input xy sheet, Make time file table input xy top. The boundary condition node set should be defined by the corresponding node selection command: Choose nodes xyz list, Choose nodes xy sheet list, Choose nodes xy top list, respectively.

## Time-field data table from regional

1. **filename** Name of the local to regional sheet mapping file (porous medium domain only).
2. **bc_time(i), bc_file(i)...end** Time [T] and filename list.

This command reads a time-file table for head [L] values from a regional model and maps them to the local model for the current set of chosen nodes. It applies to either the porous medium or overland flow domains. The time-file list should consist of head binary output files from a regional model, either `o.head_pm` for the porous medium domain or `o.head_olf` for the overland flow domain. When applied to the porous medium domain, the user must also specify a local to regional sheet mapping file (**filename**) that defines for each sheet in the local model, the corresponding sheet in the regional model to which it maps. This mapping is helpful when the local model does not have as many layers as the regional model and thus the number of node sheets are different between the two models. For example, in a local model with five sheets, the mapping file could look like:

```
1  1
2  1
3  2
4  3
5  3
```

When applied to the overland flow domain, the local to regional sheet mapping file is unnecessary and should not be supplied. At time **bc_time(i)** the values read from file **bc_file(i)** are applied and maintained until time **bc_time(i+1)**. Values read from the last file are applied until the end of the simulation.

● ● ●

#### 2.7.5.1 Interpolation

Time-varying boundary conditions change abruptly from value **bc_val(i)** to value **bc_val(i+1)** at time **bc_time(i+1)**. To have **HydroGeoSphere** use linear interpolation to smooth the values used between time panel endpoints use:

---

Interpolate

This command causes time-varying values to be interpolated between panel values for the boundary condition currently being defined. This results in a smoother application of the time-varying function.

● ● ●

#### 2.7.5.2 Scaling Factor

Sometimes, it may be desirable to scale the values in a boundary condition time series without changing the actual values entered. The following commands accomplish that, scaling the entire time series. The scaling factor can be used with any of the Time value table, Time file table, or Time raster table commands.

---

Scaling factor

1. **factor** Scaling factor [-].

Scales the values in a boundary condition time series without changing the actual values entered. For example, the following commands would cause the flux boundary condition to be scaled by a factor of two. Meaning, that from time $t = 0$ to $t = 3000$ a flux of $10^{-5}$ would be applied to the boundary condition.

```
clear chosen nodes
choose nodes top

create face set
top

boundary condition
    type
    flux

    face set
    top

    time value table
       0.0   5e-6
    3000.0   0.0
    end

    scaling factor
    2.0
end
```

•••

## Time varying scaling factor

1. **time(i), factor(i)...end** Time [T] and scaling factor [-] list.
2. **interpolate** Logical value (T/F), which if true, causes scaling factors to be interpolated from the time-value table. Otherwise, the value at the left endpoint of a time panel is used.

Scales the values in a boundary condition time series without changing the actual values entered. The scaling factor is computed from the time-value table at the current simulation time. For any simulation time that falls outside the table, the scaling factor at the endpoint nearest to that time will be used.

•••

#### 2.7.5.3 Intermittent Conditions

Boundary conditions can be turned on and off by assigning special NODATA values instead of normal input in the time-value, time-raster, and time-file tables. The default NODATA value for a time-value table is the value −99999. For time-file, time-raster, and time-field data tables the default NODATA value is the special filename "none". For example, the following code block

```
boundary condition
    type
    head

    ... etc

    time value table
       0.0       100.0
    1000.0    -99999.0
    2000.0       100.0
    end
end
```

would apply the specified head of 100 from time 0.0 to time 1000.0 and then allow the node to revert to an unconstrained condition until time 2000.0, when a specified head of 100.0 would again be applied for the duration of the simulation.

If you need to change the default NODATA value for a time-value table, then you can use the command Nodata value.

## Nodata value

1. **nodata_value** NODATA value.

Assigns a new NODATA value for a time-file table.

●●●

The following instructions may be used to change the NODATA values for rasters and files, where the input is a string variable of up to 300 characters:

Nodata raster
Nodata file

The following command can also be used to control when a boundary condition is active/inactive via a time on/off table.

## Time on/off table

1. **ton(i), toff(i)...end** Time on [T] and time off [T] table. The times should satisfy the following set of inequalities:

$$t_{\text{on}}(1) < t_{\text{off}}(1) < t_{\text{on}}(2) < t_{\text{off}}(2) < \cdots < t_{\text{on}}(n) < t_{\text{off}}(n),$$

   where $n$ is the number of panels.

This command specifies a time on/off table that defines time intervals over which a boundary condition is active. For any simulation time $t$, the boundary condition is active if and only if

$$t_{\text{on}}(i) \leq t < t_{\text{off}}(i) \text{ for some } i \in \{1, \ldots, n\}.$$

In the absence of this command, the boundary condition will be active over the duration of the simulation.

●●●

### 2.7.6 Tecplot Output

Tecplot formatted output files can be created for a specific boundary condition via the command Tecplot output.

#### Tecplot output

Causes node specific information about a boundary condition to be written a Tecplot ASCII file named *prefix*`o.Bc.`*bcname*`.dat` at each output time.

●●●

### 2.7.7 Transport

Three options are available for assigning boundary conditions to the transport solution: first-type (concentration), second-type (mass flux, concentration gradient), or third-type (Cauchy). These boundary condition are typically applied to nodes/faces located on the surface of the domain, however, first- and second-type boundary conditions can also be applied to internal nodes.

Once the transport boundary conditions have been defined, you can check them using the following instruction, which causes the current configuration to be written to the *prefix*`o.eco` file.

#### Echo transport boundary conditions

Causes the current transport boundary condition settings to be written to the *prefix*`o.eco` file.

●●●

#### 2.7.7.1 Specified Concentration

This boundary condition is also known as a first-type, Dirichlet, or constant concentration boundary condition. It is applied to nodes, so you should first choose the subset of nodes for which you want to apply the boundary condition and then issue one of the following instructions.

If a selected node was assigned a specified concentration, mass flux, or third-type value by a previous instruction, then it will not be modified by subsequent specified concentration instructions.

## Specified concentration

1. **npanel** Number of panels in the time-variable concentration function.
2. **ton_val(i), toff_val(i), bc_val(i,j), j=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], and specified concentration [M $L^{-3}$] of each species.

Chosen nodes in the currently active domain (see Section 2.8.1) are assigned a time-variable concentration value. If a node was previously assigned a specified concentration, it will remain in effect. Note that setting **bc_val(i,j)** to any value strictly less than $-100$ will cause the concentration of the $j$th solute to be unaffected by the boundary condition over the $i$th time panel.

A panel is a point in time at which the specified concentration is set to a new value. The first panel would normally start at time zero. The concentration given for the last panel will be maintained until the end of the simulation. You can assign a static concentration for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included. For example, with three panels and two species the input format is

```
3 ! npanel
ton1 toff1 bcval11 bcval12
ton2 toff2 bcval21 bcval22
ton3 toff3 bcval31 bcval32
```

●●●

## Specified concentration from file

1. **filename** Name of the file that contains the time-varying concentration information.

This file should contain the following input data:

1. **nnde** Number of nodes to be assigned concentration data.
2. **npanel** Number of panels in the time-variable concentration function.
3. **ton_val(i), toff_val(i), i=1,npanel** Time on [T] and time off [T] list.
4. For each node **k=1,nnde**:
   (a) **node(k)** Node number.
   (b) **bc_val(i,j,k), i=1,npanel, j=1,nspeciesmob** Specified concentration [M $L^{-3}$] history for each species.

For example, with two nodes, three panels, and two species the file format is

```
2 ! nnde
3 ! npanel
ton1 toff1
ton2 toff2
ton3 toff3
n1
bcval111 bcval211 bcval311 ! concentrations for species 1
bcval121 bcval221 bcval321 ! concentrations for species 2
n2
bcval112 bcval212 bcval312 ! concentrations for species 1
bcval122 bcval222 bcval222 ! concentrations for species 2
```

Listed nodes are assigned the time-variable concentration values contained in the file. If a node was previously assigned a specified concentration it will remain in effect. Note that setting **bc_val(i,j,k)** to any value strictly less than $-100$ will cause the concentration of the $j$th solute at the $k$th node to be unaffected by the boundary condition over the $i$th time panel.

Although all nodes in the file share the same time on/time off panel information the concentration values in each panel can vary from node to node.

● ● ●

#### 2.7.7.2 Specified Mass Flux

This boundary condition is also known as a second-type, Neumann, or constant mass flux boundary condition. It is applied to nodes, so you should first choose the subset of nodes for which you want to apply the boundary condition.

If a selected node was assigned a specified concentration or third-type concentration by a previous instruction, then it will not be modified by subsequent specified mass flux instructions.

If a selected node was assigned a specified mass flux value by a previous instruction, then mass fluxes assigned by subsequent instructions will be cumulative.

## Specified mass flux

1. **npanel** Number of panels in the time-variable mass flux function.
2. **ton_val(i), toff_val(i), bc_val(i,j), j=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], and specified mass flux [M $T^{-1}$] of each species.

Chosen nodes in the currently active domain (see Section 2.8.1) are assigned a time-variable mass flux value. This is a passive injection of solute mass which has no effect on the flow solution. If a node was previously assigned a first or third-type concentration, it will remain in effect.

A panel is a point in time at which the specified mass flux is set to a new value. The first panel would normally start at time zero. The mass flux given for the last panel will be maintained until the end of the simulation. You can assign a static mass flux for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that the mass flux values are per unit time and the total mass which will be injected for a given timestep can be calculated by multiplying the value given here by the timestep length and the number of chosen nodes.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included. For example, with three panels and two species the input format is

```
3 ! npanel
ton1 toff1 bcval11 bcval12
ton2 toff2 bcval21 bcval22
ton3 toff3 bcval31 bcval32
```

• • •

## Interpolate mass flux

This command causes time-varying mass fluxes to be interpolated between panel values, which results in a smoother application of the mass flux function.

• • •

#### 2.7.7.3 Specified Third-Type Concentration

This boundary condition is also known as a third-type or Cauchy boundary condition. It is applied to faces, so you should first choose the subset of faces for which you want to apply the boundary condition.

If a node belonging to a selected face was assigned a specified concentration by a previous instruction, then it will not be modified by subsequent third-type concentration instructions.

If a node belonging to a selected face was assigned a third-type concentration value by a previous instruction, then third-type concentration fluxes assigned in subsequent instructions will be cumulative. This is because third-type concentrations are applied to faces and any node common to two such faces requires a contribution from each face.

## Specified third-type concentration

1. **calcflux** Logical value (T/F) that determines how fluid fluxes are handled for this third-type boundary condition. If false, then read the following:
   (a) **userflux** Fluid flux value [L $T^{-1}$].
2. **npanel** Number of panels in the time-variable concentration function.
3. **ton_val(i), toff_val(i), bc_val(i,j), j=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], and specified concentration [M $L^{-3}$] of each species.

Chosen faces in the currently active domain (see Section 2.8.1) are assigned a time-variable third-type concentration value unless they were previously assigned a first-type concentration.

If the variable **calcflux** is true, fluxes are calculated by **HydroGeoSphere** from the flow solution and used to calculate the third-type boundary condition. Only positive fluxes (i.e., flowing into domain) are used. Otherwise, **HydroGeoSphere** reads a flux value, **userflux**, which is used instead.

A panel is a point in time at which the specified third-type concentration is set to a new value. The first panel would normally start at time zero. The concentration given for the last panel will be maintained until the end of the simulation. You can assign a static concentration for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0, and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included. For example, if **calcflux** is false, the number of panels is three, and the number of species is two, then the input format is

```
F
userflux
3 ! npanel
ton1 toff1 bcval11 bcval12
ton2 toff2 bcval21 bcval22
ton3 toff3 bcval31 bcval32
```

● ● ●

## Specified third-type concentration from file

1. **filename** Name of the file that contains the time-varying third-type concentration information.

This file should contain the following input data:

1. **nelem** Number of elements.
2. **npanel** Number of panels in the time-variable concentration function.
3. **calcflux** Logical value (T/F) that determines how fluid fluxes are handled for this third-type boundary condition. If false, then read the following:
   (a) **userflux** Fluid flux value [L $T^{-1}$].
4. **ton_val(i), toff_val(i), i=1,npanel** Time on [T] and time off [T] list.
5. For each element **k=1,nelem**:
   (a) **nel(k), n1(k), n2(k), n3(k), n4(k)** Element number and four node numbers defining a face.
   (b) **bc_val(i,j,k), i=1,npanel, j=1,nspeciesmob** Third-type concentration [M $L^{-3}$] history for each species.

For example, if the number of elements is two, the number of panels is three, the number of species is two, and **calcflux** is false, then the input format is

```
2 ! nelem
3 ! npanel
F
userflux
ton1 toff1
ton2 toff2
ton3 toff3
nel1 n1(1) n2(1) n3(1) n4(1)
bcval111 bcval211 bcval311 ! concentrations for species 1
bcval121 bcval221 bcval321 ! concentrations for species 2
nel2 n1(2) n2(2) n3(2) n4(2)
bcval112 bcval212 bcval312 ! concentrations for species 1
bcval122 bcval222 bcval322 ! concentrations for species 2
```

Faces defined by the four nodes in the listed elements in the currently active domain (see Section 2.8.1) are assigned a time-variable third-type concentration value unless they were previously assigned a first-type concentration. It is the responsibility of the user to ensure that the nodes define a face that is on the exterior of the domain.

For three-node faces, enter a value of zero for node **n4**.

If the variable **calcflux** is true, fluxes are calculated by **HydroGeoSphere** from the flow solution and used to calculate the third-type boundary condition. Only positive fluxes (i.e., flowing into domain) are used. Otherwise, **HydroGeoSphere** reads a flux value, **userflux**, which is used instead.

If **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

Although all elements in the file share the same time on/time off panel information the concentration values in each panel can vary from element to element.

● ● ●

## Specified third-type concentration from face file

1. **filename** Name of the file that contains the time-varying third-type concentration information.

This file should contain the following input data:

1. **nface** Number of faces.
2. **npanel** Number of panels in the time-variable concentration function.
3. **calcflux** Logical value (T/F) that determines how fluid fluxes are handled for this third-type boundary condition. If false, then read the following:
   (a) **userflux** Fluid flux value [L $T^{-1}$].
4. **ton_val(i), toff_val(i), i=1,npanel** Time on [T] and time off [T] list.
5. For each face **k=1,nface**:
   (a) **nfce(k)** Face number.
   (b) **bc_val(i,j,k), i=1,npanel, j=1,nspeciesmob** Third-type concentration [M $L^{-3}$] history for each species.

For example, if the number of faces is two, the number of panels is three, the number of species is two, and **calcflux** is false, then the input format is

```
2 ! nface
3 ! npanel
F
userflux
ton1 toff1
ton2 toff2
```

```
ton3 toff3
nfce1
bcval111 bcval211 bcval311 ! concentrations for species 1
bcval121 bcval221 bcval321 ! concentrations for species 2
nfce2
bcval112 bcval212 bcval312 ! concentrations for species 1
bcval122 bcval222 bcval322 ! concentrations for species 2
```

Listed faces in the currently active domain (see Section 2.8.1) are assigned a time-variable third-type concentration value unless they were previously assigned a first-type concentration. It is the responsibility of the user to ensure that the faces belong to the exterior of the domain.

If the variable **calcflux** is true, fluxes are calculated by **HydroGeoSphere** from the flow solution and used to calculate the third-type boundary condition. Only positive fluxes (i.e., flowing into domain) are used. Otherwise, **HydroGeoSphere** reads a flux value, **userflux**, which is used instead.

If **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

Although all faces in the file share the same time on/time off panel information the concentration values in each panel can vary from face to face.

•••

## Specified third-type flushing

1. **npanel** Number of panels in the time-variable fluid flux function.
2. **ton_val(i), toff_val(i), q_val(i), i=1,npanel** Time on [T], time off [T], and flux per unit area [L T$^{-1}$] list.

A flushing boundary condition $q(t)C$ is applied using the time series of $q$ for chosen faces.

•••

#### 2.7.7.4 Thermal Energy

This boundary condition is also known as a second-type, Neumann, or constant temperature flux boundary condition. It is applied to nodes, so you should first choose the subset of nodes for which you want to apply the boundary condition.

If a selected node was assigned a specified concentration or third-type concentration by a previous instruction, then it will not be modified by subsequent specified temperature flux instructions.

If a selected node was assigned a specified temperature flux value by a previous instruction, then temperature fluxes assigned by subsequent instructions will be cumulative.

## Specified temperature flux

1. **npanel** Number of panels in the time-variable, specified temperature flux function.
2. **ton_val(i), toff_val(i), bc_val(i), bc_temp(i), i=1,npanel** Time on [T], time off [T], specified volume flux (of liquid) [L$^3$ T$^{-1}$] multiplied by the heat capacity [J kg$^{-1}$ K$^{-1}$] and density [M L$^{-3}$] of the boundary medium, and temperature of water entering [K] list.

Chosen nodes that belong to the currently active domain (see Section 2.8.1) are assigned a time-variable temperature flux value.

Although a fluid volume flux **bc_val** is specified, it does not influence the flow solution in any way. It is merely intended to give the user a straightforward way to input a known amount of energy to the system, as a function of fluid volume and temperature.

•••

The following instructions may be used to define the atmospheric inputs for thermal energy transport discussed in Section 2.11.7.1 and summarized in Equation 2.169.

## Atmosphere...End

**grok** reads instructions that define atmospheric input parameters until it encounters an End instruction.

•••

## Incoming shortwave radiation

1. **npanel** Number of panels in the time-variable, incoming shortwave radiation function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and incoming shortwave radiation [M T$^{-3}$] ($K^{\downarrow}$ in Equation 2.170). Default value is $1.10 \times 10^2$ J m$^{-2}$ s$^{-1}$.

•••

## Sinusoidal incoming shortwave radiation

1. **npanel** Number of panels in the time-variable, sinusoidal incoming shortwave radiation function.
2. **ton_val(i), value(i), amp(i), phase(i), period(i), i=1,npanel** Time on [T], vertical shift [M T$^{-3}$] (mid-point incoming shortwave radiation, $K^{\downarrow}$ in Equation 2.170), amplitude [M T$^{-3}$], phase [-], and period [T].

• • •

## Incoming shortwave radiation for chosen faces

1. **set_name** Name of the face set, up to 40 characters.
2. **time(i), value(i)...end** Time on [T] and incoming shortwave radiation [M T$^{-3}$] ($K^{\downarrow}$ in Equation 2.170).

Applies the incoming shortwave radiation timeseries to the faces in the face set that belong to the surface of the model domain. The face set must be defined by the commands Create face set or Create face set from chosen faces. This command takes precedence over the commands Incoming shortwave radiation and Sinusoidal incoming shortwave radiation for the set of selected faces, whereas these commands apply to any unselected faces in the surface of the model domain. In the event that this command is issued more than once with common faces, the timeseries corresponding to the most recent issuing of the command will be honoured. The following example illustrates how to use this command.

```
clear chosen faces
choose faces top

create face set from chosen faces
top

atmosphere
    incoming shortwave radiation for chosen faces
    top
    0.0 110.0
    end
end
```

Note that all face selection and set creation must occur outside the Atmosphere...End command block.

• • •

## Cloud cover

1. **npanel** Number of panels in the time-variable, cloud cover function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and cloud cover [-] ($C_c$ in Equation 2.172). Default value is 0.5.

• • •

## Incoming longwave radiation

1. **npanel** Number of panels in the time-variable, incoming longwave radiation function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and incoming longwave radiation [M T$^{-3}$] ($L^{\downarrow}$ in Equation 2.173). Default value is $3.0 \times 10^2$ J m$^{-2}$ s$^{-1}$.

•••

## Temperature of air

1. **npanel** Number of panels in the time-variable, air temperature function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and air temperature [°C] ($T_a$ in Equation 2.174). Default value is 15 °C.

•••

## Sinusoidal temperature of air

1. **npanel** Number of panels in the time-variable, sinusoidal air temperature function.
2. **ton_val(i), value(i), amp(i), phase(i), period(i), i=1,npanel** Time on [T], vertical shift [°C] (mid-point temperature, $T_a$ in Equation 2.174), amplitude [°C], phase [-], and period [T].

•••

These instructions are used to define the sensible heat flux $Q_h$ in Equation 2.176 and latent heat flux $Q_E$ in Equation 2.177.

## Density of air

1. **npanel** Number of panels in the time-variable, air density function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and density of air [M L$^{-3}$] ($\rho_a$ in Equation 2.176). Default value is 1.225 kg m$^{-3}$.

•••

## Specific heat of air

1. **npanel** Number of panels in the time-variable, air specific heat function.

2. **ton_val(i), value(i), i=1,npanel** Time on [T] and specific heat of air [L$^2$ T$^{-2}$ K$^{-1}$] ($c_a$ in Equation 2.176). Default value is $7.17 \times 10^2$ J kg$^{-1}$ K$^{-1}$.

•••

## Wind speed

1. **npanel** Number of panels in the time-variable, wind speed function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and wind speed [L T$^{-1}$] ($V_a$ in Equation 2.176). Default value is 1.0 m s$^{-1}$.

•••

## Sinusoidal wind speed

1. **npanel** Number of panels in the time-variable, sinusoidal wind speed function.
2. **ton_val(i), value(i), amp(i), phase(i), period(i), i=1,npanel** Time on [T], vertical shift [L T$^{-1}$] (mid-point wind speed, $V_a$ in Equation 2.176), amplitude [L T$^{-1}$], phase [-], and period [T].

•••

## Drag coefficient

1. **npanel** Number of panels in the time-variable, drag coefficient function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and drag coefficient [-] ($c_D$ in Equation 2.176). Default value is $2.0 \times 10^{-3}$.

•••

## Latent heat of vapourization

1. **npanel** Number of panels in the time-variable, latent heat of vapourization function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and latent heat of vapourization [L$^2$ T$^{-2}$] ($L_V$ in Equation 2.177). Default value is $2.258 \times 10^6$ J kg$^{-1}$.

•••

## Specific humidity of air

1. **npanel** Number of panels in the time-variable, air specific humidity function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and specific humidity of air [M M$^{-1}$] ($SH_a$ in Equation 2.177). Default value is 0.01062.

●●●

## Soil-Water suction at surface

1. **npanel** Number of panels in the time-variable, soil-water suction at surface function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and soil-water suction at surface [L] ($\psi_g$ in Equation 2.179). Default value is 0.138 m.

●●●

## Saturation vapour pressure

1. **npanel** Number of panels in the time-variable, saturation vapour pressure function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and saturation vapour pressure [M L$^{-1}$ T$^{-2}$] ($e_{sat}[T_g]$ in Equation 2.180). Default value is $1.704 \times 10^3$ Pa.

●●●

## Relative humidity

1. **npanel** Number of panels in the time-variable, relative humidity function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and relative humidity [-]. Default value is 0.75.

●●●

## Pressure of air

1. **npanel** Number of panels in the time-variable, air pressure function.
2. **ton_val(i), value(i), i=1,npanel** Time on [T] and air pressure [M L$^{-1}$ T$^{-2}$] ($p_a$ in Equation 2.180). Default value is $1.013 \times 10^5$ Pa.

●●●

#### 2.7.7.5 Immiscible Phase Dissolution Source

An immiscible phase dissolution source is one in which it is assumed that there is dissolution of an immobile liquid or solid phase into the subsurface water until all dissolvable material is exhausted. The following instruction can be used to define the source:

---

## Immiscible phase dissolution data

1. **diss_mass(i), i=1,nspeciesmob** Mass of dissolvable immiscible phase per unit volume of porous medium [M $L^{-3}$] for each solute.
2. **diss_conc(i), i=1,nspeciesmob** Aqueous solubility for the immiscible phase expressed as the mass of solute per unit volume of aqueous solution [M $L^{-3}$] for each solute.
3. **diss_decay(i), i=1,nspeciesmob** First-order decay constant [$T^{-1}$] for dissolvable immiscible phase.

This instruction can be used to simulate an immiscible phase that releases a species into solution by dissolution.

Chosen nodes are assigned first-type concentration values equal to **diss_conc**. It is assumed that the total mass of the immiscible phase is located in the matrix only. For each time step, the remaining mass of the immiscible phase is updated by subtracting the mass released in solution for the time step, which is calculated by multiplying the contributing total volume (from all shared elements) by the porosity and the variable **diss_mass**. The decay constant is applied to the immiscible phase and is used to update its remaining mass.

When the immiscible phase associated with a node is entirely dissolved and the remaining immiscible phase mass reaches zero, the node reverts back to an unconstrained condition for transport. Care has to be taken to avoid locating dissolution nodes on inflow boundaries of the model because, once nodes are unconstrained, there will be an artificial advective flux entering the domain if the nodal concentration is greater than zero.

• • •

#### 2.7.7.6 Zero-Order Source

A zero-order source is one in which the medium itself produces a solute. For example, some soils produce radon gas as a result of radioactive decay.

---

## Zero order source

1. **npanel** Number of panels in the time-variable, zero-order source function.
2. **ton(i), toff(i), prate(i,j), j=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], solute mass production rate [M $L^{-3}$ $T^{-1}$].

Nodes that belong to the chosen zones are assigned zero-order source boundary conditions.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term specified for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to one, **ton** to zero, and **toff** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **prate** should be included. For example, with three panels and two species the input format is

```
3 ! npanel
ton1 toff1 prate11 prate12
ton2 toff2 prate21 prate22
ton3 toff3 prate31 prate32
```

•••

## Zero order source with partitioning

1. **npanel** Number of time panels.
2. **ton(i), toff(i), prate(i,j), j=1,nspeciesmob, pcoeff(i,k), k=1,nspeciesmob, i=1,npanel** Time on [T], time off [T], solute mass production rate at fully saturated conditions ($P_s$) [M L$^{-3}$ T$^{-1}$], aqueous/gas phase partitioning coefficient ($H_{cc}$) [-].

Nodes that belong to the chosen zones are assigned zero-order source boundary conditions. The effective production rate $P$ is defined by the following equation that was derived via a simple instant-equilibration mass-balance model

$$P = S_w \frac{H_{cc} \cdot P_s}{1 + (H_{cc} - 1)S_w},$$

where $S_w$ is the water saturation [-] and the partitioning coefficient, $H_{cc}$, is defined as the ratio of solute concentration in the aqueous phase to solute concentration in the gas phase. This command is equivalent to the command Zero order source in the limit as $H_{cc} \to \infty$ (i.e., no partitioning) and for $S_w = 1$ regardless of $H_{cc}$. The unpartitioned formulation that sets $P = P_s$ can be obtained for a given time panel/species by setting the partitioning coefficient to any strictly negative value for that time panel/species.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term specified for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to one, **ton** to zero, and **toff** to a large number.

Note that if **nspeciesmob** is greater than one, then additional values of **prate** and **pcoeff** should be included. For example, with three panels and two species the input format is

```
3 ! npanel
ton1 toff1 prate11 prate12 pcoeff11 pcoeff12
ton2 toff2 prate21 prate22 pcoeff21 pcoeff22
ton3 toff3 prate31 prate32 pcoeff31 pcoeff32
```

•••

### Zero order source saturation threshold

1. **threshold(i), i=1,nspeciesmob** Water saturation threshold [-].

This command has no effect unless it is used in conjunction with the commands Zero order source or Zero order source with partitioning. For each species, it specifies a nodal water saturation threshold for the porous media domain. Only those nodes whose water saturation is greater than or equal to the threshold may act as zero-order source nodes. By default the threshold value is zero for all species. Hence, any nodes selected by the commands Zero order source or Zero order source with partitioning always act as zero-order source nodes.

•••

## 2.8 Materials and Material Properties

### 2.8.1 General

Currently, the following eight domains can be defined in **HydroGeoSphere**:

1. Porous media
2. Dual continua
3. Discretely-fractured
4. Surface flow
5. ET
6. Channel flow
7. Well
8. Tile

Porous media and dual continua domains are defined by three-dimensional 8-node (block) or 6-node (prism) elements, discretely-fractured, surface flow, and ET domains are defined by two-dimensional 4-node (rectangle) or 3-node (triangle) elements, and well, tile, and channel flow domains are defined by 1-node (line) elements. By default, every 3-D element in the problem domain is a porous media element. Elements of the other domain types may or may not be defined for a specific problem.

Each porous media element is assigned a zone (i.e., material) number during grid generation. In simple cases, all elements are assigned a zone number of 1, while in more complex cases, elements may have been assigned different zone numbers. For example, if a multilayered grid was generated using the instruction Generate layers interactive...End then the elements would be assigned zone numbers based on the layer number (i.e., elements in the lowest layer number 1 would be assigned a material number of 1).

By default, all zones, and therefore all elements in the domain, are assigned the same default porous media properties, which are listed in Table 2.6. These values are fixed by **HydroGeoSphere** and cannot be modified by the user. However, there are other ways of changing the porous media zone properties as we will discuss below.

The first step in modifying zoned properties for a given problem is to indicate which type of medium is to be manipulated. The following instruction does this:

## Use domain type

1. **zone_type** Can be one of the strings: "porous media", "dual", "surface", "fracture", "channel", "well", "tile", or "et".

Causes **grok** to read a character string defining the type of domain to which additional instructions should be applied.

•••

## Initialize domain

1. **zone_type** Can be one of the strings: "porous media", "dual", "surface", "fracture", "channel", "well", "tile", or "et".

This command initializes a domain of type **zone_type** for the current set of chosen elements/faces/segments (depending on the domain), assigns all selected elements/faces/segments to zone number one, and sets default property values for that zone. It is intended as a convenience command when defining a domain. For example, when defining the overland flow domain, the usual approach is

```
use domain type
surface

clear chosen faces
choose faces top
```

```
    new zone
    1
```

With this command, defining the overland flow domain is simplified to

```
    clear chosen faces
    choose faces top

    initialize domain
    surface
```

A benefit of using this command is that it avoids the common mistake of forgetting to issue the new zone command at least once at the start of the domain definition.

●●●

## Initialize default surface domain

This command initializes a surface domain for all faces in the top node sheet, assigns those faces to zone number one, and sets default property values for that zone.

●●●

## Initialize default et domain

This command initializes an ET domain for all faces in the top node sheet, assigns those faces to zone number one, and sets default property values for that zone.

●●●

**Important:** To ensure that the internal node mapping arrays used by **grok** are set up correctly, we strongly recommend that domains be defined in the following order for multi-domain models: porous media, dual, surface, fracture, channel, well, tile, ET.

Models with many zones may write a large amount of output to the `.eco` file causing it to balloon in size, potentially slowing down **grok** execution due to excessive I/O operations. The following command is designed to address this problem.

## Limit zone output

1. **max_zone** Maximum zone number to output.

This command causes **grok** to limit the output of some zonal information (e.g., material properties) to the `.eco` file for all domain types. Output is written for all zone numbers between one and at most **max_zone**. Setting **max_zone** = 0 will truncate all output. By default zone output to the `.eco` file is unlimited.

●●●

#### 2.8.1.1 Defining a New Zone

In order to define a new zone, elements of the proper type must first be chosen using the instructions given in Section 2.4. For example, 3-D block elements must first be selected before a new porous media or dual zone can be defined, 2-D rectangular faces are selected for fracture, surface or ET zones, and 1-D segments are selected for channel, well, or tile zones.

In a problem where there are to be dual, fracture, surface, channel, well, tile, or ET zone types, a new zone must be defined since by default there are no elements of these types after grid generation. This is not the case for porous media zones, where by default all 3-D elements are in porous media zone 1.

Once elements of the appropriate type have been chosen, the following command groups them into a single zone.

---

## New zone

1. **num_zone** Zone number.

Chosen elements are assigned a new zone number in the currently active domain. If **num_zone** is greater than the total number of zones for the currently active domain, the total number of zones will be incremented and default properties for that domain will be assigned.

•••

The following set of instructions, inserted in the *prefix*.grok file would create a new fracture zone.

```
use domain type
fracture

clear chosen faces
choose faces z plane
0.05
1.e-5

new zone
1
```

The following command was added to support models with complex zone assignments. Zero is not a valid zone number. If all elements are first assigned to zone zero, then it is possible to easily find those elements that were incorrectly assigned a valid zone number.

## Assign zone zero

Assign all elements to zone number zero.

• • •

#### 2.8.1.2 Saving and Retrieving Element Zone Numbers

The following commands can be used to store or retrieve porous media, surface, or ET domain element zone numbers.

## Write zones to file

1. **zone_file** Name of the file to which element zone numbers will be written.

Writes element zone numbers to file for the porous media domain. Each line of the file contains the element number followed by the assigned zone number.

• • •

## Read zones from file

1. **zone_file** Name of the file from which zone information will be read.

Reads element zone numbers for the porous media domain. Each line of the file contains the porous media element number followed by the zone number. For example, if you want porous media elements 1, 5, 9, and 44 to have zone number 2 and element 8 to have zone number 3, then the file would contain:

```
 1  2
 5  2
 9  2
44  2
 8  3
```

• • •

## Read overland zones from file

1. **zone_file** Name of the file from which zone information will be read.

Reads element zone numbers for the overland flow domain. Each line of the file contains the porous media element number that corresponds to an overland flow element followed by the zone number.

•••

## Read et zones from file

1. **zone_file** Name of the file from which zone information will be read.

Reads element zone numbers for the ET domain. Each line of the file contains the porous media element number that corresponds to an ET element followed by the zone number.

•••

## Read pm zones from voxet file

1. **filename** Filename of voxet file (exported to ASCII format), up to 256 characters. Note that the length units in this file must match the model's length units.
2. **colnum** Column number in voxet file where zone numbers are found. An integer value $\geq 4$ (the first three columns are reserved for $xyz$-coordinates of voxel centers).
3. **nearest** Logical value (T/F) that defines how the lookup behaves if the query point is not found. If true, the zone number of the voxel nearest to the query point is used. Otherwise, an error is raised.

This command assigns porous media material zone information from a GoCAD voxet file that has been exported to a simplified ASCII format, an example of which is given below. Note that this command applies only to triangular prism or hexahedral block meshes.

```
# type: Voxet
# domain: Depth
# nu: 121
# nv: 191
# nw: 79
# step_u: 25 0 0
# step_v: 0 25 0
# step_w: 0 0 -25
# origin: 313150 6071000 400
X Y Z Zone_Numbers
313150 6071000 400 1
313175 6071000 400 1
313200 6071000 400 2
313225 6071000 400 3
313250 6071000 400 3
313275 6071000 400 3
```

```
313300 6071000 400 3
   :       :     :  :
   :       :     :  :
```

●●●

## Read zones from raster

1. **zone_raster, (band)** Name of raster file containing zone numbers and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).

Assigns zone numbers to chosen elements in the porous medium domain or to chosen faces in the overland flow or ET domains from the specified raster band. For each chosen element/face, the zone number is assigned from the raster cell that contains the 2-D centroid of that element/face. It is important to remember that you must issue the New zone command at least once prior to issuing this command when assigning zone numbers to either the overland flow or ET domains. For example:

```
use domain type
surface

clear chosen faces
choose faces top

new zone
1

read zones from raster
olf_zones.asc
```

●●●

## Read zones from raster, dominant class

1. **zone_raster, (band)** Name of raster file containing zone numbers and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).

Assigns zone numbers to chosen elements in the porous medium domain or to chosen faces in the overland flow or ET domains from the specified raster band. For each chosen element/face, the zone number is assigned from the raster cell that has the largest area of intersection with the projection of that element/face to the *xy*-plane. It is important to remember that you must issue the New zone command at least once prior to issuing this command when assigning zone numbers to either the overland flow or ET domains.

• • •

## Map zones from raster

1. **zone_raster, (band)** Name of raster file and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).
2. **rval(i), zone(i)...end** Raster value to zone number mapping.

Assigns zone numbers to chosen elements in the porous medium domain or to chosen faces in the overland flow or ET domains from the specified raster band. For each chosen element/face, the zone number is assigned by the raster value to zone number mapping from the raster cell that contains the 2-D centroid of that element/face. If a raster value does not belong to the specified mapping, then the zone number for that element/face is not updated. It is important to remember that you must issue the New zone command at least once prior to issuing this command when assigning zone numbers to either the overland flow or ET domains. For example:

```
use domain type
surface

clear chosen faces
choose faces top

new zone
1

map zones from raster
olf_zones.asc
1 1
2 2
5 3
8 4
9 5
end
```

In the above example, raster value 1 corresponds to zone 1, raster value 2 corresponds to zone 2, raster value 5 corresponds to zone 3, and so on.

• • •

## Map zones from raster, dominant class

1. **zone_raster, (band)** Name of raster file and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).
2. **rval(i), zone(i)...end** Raster value to zone number mapping.

Assigns zone numbers to chosen elements in the porous medium domain or to chosen faces in the overland flow or ET domains from the specified raster band. For each chosen element/face, the zone number is assigned by the raster value to zone number mapping from the raster cell that has the largest area of intersection with the projection of that element/face to the *xy*-plane. If a raster value does not belong to the specified mapping, then the zone number for that element/face is not updated. It is important to remember that you must issue the New zone command at least once prior to issuing this command when assigning zone numbers to either the overland flow or ET domains.

• • •

## Read zones from csv point cloud

1. **filename** Name of CSV file.
2. **nskip** Number of lines to skip from the start of the input file. For example, if you need to skip the header line.
3. **zone_default** Default zone number that is assigned when there is a tie in zone number voting. Note that by specifying a negative value, an element's zone number is unchanged in the case of a tie.

Assigns zone numbers to elements in the porous medium domain from a point cloud. Each line of the input file contains the *xyz*-coordinates of a point followed by a zone number. For each point in the file, the element whose centroid is nearest to that point receives a “vote” for the corresponding zone number. The zone number that receives the most votes is then assigned to that element. In the case of a tie, the element is either assigned the user specified default zone number (if **zone_default** $\geq 0$), or its zone number is unchanged (if **zone_default** $< 0$). If all elements are initially assigned a positive zone number, then you may consider setting the default zone number to zero to see where any ties have occurred.

• • •

## Read zones from shapefile for chosen elements

1. **filename** Name of the shapefile without the file extension.
2. **nodata_replace** Zone number to assign an element that does not belong to any polygon.

3. **zone_attribute** Field name (up to 11 characters) used to assign the zones, which must be written exactly as in the `.dbf` file (case sensitive).

4. **zone_offset** A nonnegative integer that is added to zone numbers read from the shapefile. Input a value of zero to use the original zone numbers from the file.

Chosen elements (porous media domain) or chosen faces (surface domain) whose centroids lie within a polygon defined by the shapefile are assigned a zone number from the specified zone attribute. Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.* The variable **zone_offset** can be used to preserve existing zone numbers. For example, if a model already has zones defined that are numbered from 1 to 7, and the shapefile contains zones numbered from 1 to 4, you can set **zone_offset** to 7, and zones assigned from the shapefile will be numbered from 8 to 11.

●●●

#### 2.8.1.3 Selecting Zones

These commands can be used to alter the set of chosen zones for the currently active domain.

### Clear chosen zones

All zones in the currently active domain are flagged as *not* chosen. This command is recommended if you are unsure of which zones are chosen due to previously issued commands.

●●●

### Choose zones all

All zones in the currently active domain are chosen. This command is useful if you wish to assign a property to all defined zones.

●●●

### Choose zone number

1. **num_zone** Zone number.

The zone numbered **num_zone** in the currently active domain is chosen.

●●●

### Choose zones list

1. **filename** Name of file that contains the list of zone numbers.

All zones listed in the file in the currently active domain are chosen. The input file consists of a list of zone numbers, one entry per line. The file is read until end-of-file is reached.

● ● ●

#### 2.8.1.4 Modifying Zoned Properties

There are a number of instructions which can be used to modify the property values associated with a zone or group of zones. Before these instructions are issued, it is necessary to select the appropriate type of media and then choose the zones which you want to modify.

For example, you could assign a new porous media hydraulic conductivity to all zones, and thus to all elements in the problem, with the following set of instructions inserted in the *prefix*`.grok` file:

```
use domain type
porous media

clear chosen zones

choose zones all

k isotropic
1.0e-5
```

In this case, we are applying the instruction K isotropic to zones of the porous media domain, although it is equally valid to use it with the dual continua domain. However, if you try to use this instruction with zones of type fracture or surface, for example, warning messages will be issued to the screen and *prefix*`o.eco` file, and execution will halt.

The instructions which are valid in specific situations are discussed in the relevant sections of the manual. For example, instructions which can be used for defining saturated flow properties are described in Section 2.8.2.

Another way to define zone properties is through the use of a material properties file, which should be located in the same directory as the *prefix*`.grok` file. This file contains lists of media-specific instructions that can be used to define properties for one or more named materials. These material properties can then be assigned to the current set of chosen zones. To assign new properties through the use of a material properties file, we first need to issue the following instruction:

## Properties file

1. **filename** Name of the material properties file, up to 256 characters.

This file will be searched for materials given as input to the Read properties instruction described below.

•••

The Properties file instruction has two benefits: it allows you to create sets of material properties and give them meaningful file names, and it allows you to easily switch between material property sets merely by changing the file name given in the *prefix*.grok file.

Any line in a material properties file that is completely blank or that begins with an exclamation point (!) is treated as a comment and is ignored by **grok**. This allows you to include comments whenever required.

Each distinct material in the file is identified by a unique label and may contain instructions which are to be applied to the current zone type. For example, instructions which can be used for defining porous media properties when simulating saturated flow (as described in Section 2.8.2.1) may be included. Figure 2.10 shows an example of a material defined for the verification problem discussed in Section 1.5.1.

```
!-----------------------------------------
Porous medium

k isotropic
500.0

specific storage
0.0

porosity
1.0

longitudinal dispersivity
10.0

transverse dispersivity
0.1

transverse vertical dispersivity
0.1

tortuosity
0.1

end material
```

Figure 2.10: Example of a porous media domain property file *prefix*.mprops.

To make use of the material properties file, you would issue the following instruction:

## Read properties

1. **mat_name** Name of the material, up to 40 characters.

Chosen zones are assigned properties of the material named **mat_name** in the current material properties file as defined in a Properties file instruction.

●●●

So for example, the following set of instructions could be inserted in the *prefix*.grok file:

```
use domain type
porous media

properties file
my.mprops

clear chosen zones

choose zones all

read properties
sand
```

The instruction Read properties would, in this case, search the porous media material properties file my.mprops for a material named sand. If found, it would read the instructions defining the material and modify the porous media properties for the current set of chosen zones.

A summary of the final data which has been defined for each zone is listed in the *prefix*o.eco file, and an example is shown in Figure 2.11.

```
-----------------------------------------------------------
POROUS MEDIA PROPERTIES

ZONE:  1
MATERIAL: porous medium
Consists of          100  elements out of          100
Kxx:   500.000
Kyy:   500.000
Kzz:   500.000
Specific storage:   0.00000
Porosity:   1.00000
```

```
Longitudinal dispersivity    10.0000
Transverse dispersivity   0.100000
Transverse vertical dispersivity   0.100000
Tortuosity   0.100000
Bulk density    2650.00
Immobile zone porosity    0.00000
Mass transfer coefficient    0.00000
   100  elements of     100  have been assigned properties
```

Figure 2.11: Example output for a porous media material.

In this example, because flow is saturated, no variably-saturated porous media flow properties need to be defined in the material properties file. Also, default values for properties (e.g., bulk density, immobile zone porosity, etc.) that have not been modified in the *prefix*`.grok` or material properties file are used.

## 2.8.2 Saturated Subsurface Flow

### 2.8.2.1 Porous Medium

**HydroGeoSphere** is designed to perform the flow simulation in saturated mode unless instructed otherwise, and unless you modify the default values, all zones (and elements) in the domain will be assigned the default porous media properties which are listed in Table 2.6. These values are representative of a sand.

Note that the default state of the hydraulic conductivity tensor (**K** in Equation 2.2) is that it is isotropic and that all off-diagonal terms are zero.

You can use the general methods and instructions outlined in Section 2.8.1 to modify the default distribution of saturated porous media properties. A general porous medium layout is shown as the following instruction:

```
use domain type
porous media

properties file
    {props_file_name.mprops}

clear chosen nodes/elements/segments/faces/faces by nodes/zones
    {choose nodes/elements/segments/faces/faces by nodes/zones}
    {choose_description}

new zone
```

```
        {zone_type}

    clear chosen zones
    choose zone number
        {num_zone} ! same as {zone_type}

    read properties
        {mat_name}
```

Table 2.6: Default values for porous media saturated flow properties.

| Parameter | Value | Unit |
|---|---|---|
| Name | Default Sand | |
| Hydraulic conductivity terms: | | |
| $K_{xx}$ | $7.438\times10^{-5}$ | m s$^{-1}$ |
| $K_{yy}$ | $7.438\times10^{-5}$ | m s$^{-1}$ |
| $K_{zz}$ | $7.438\times10^{-5}$ | m s$^{-1}$ |
| $K_{xy}$ | 0.0 | m s$^{-1}$ |
| $K_{xz}$ | 0.0 | m s$^{-1}$ |
| $K_{yz}$ | 0.0 | m s$^{-1}$ |
| Specific storage $S_s$ | $1.0\times10^{-4}$ | m$^{-1}$ |
| Porosity $\theta_s$ | 0.375 | |
| Poisson's ratio $\nu^*$ | 0.3 | |
| Solids compressibility $\gamma_s$ | 0.0 | kg$^{-1}$ m s$^2$ |
| Unsaturated flow relation type | Pseudo-soil | |

Note that each instruction given here has an associated scope of operation. For example, some can only be used in the *prefix*`.grok` file, in which case they will affect the current set of chosen zones or elements. Other instructions can only be used in a porous media properties (`.mprops`) file, in which case they affect only the named material of which they are a part. Finally, some instructions can be used in both types of files, in which case their behaviour will be as described above, and will depend on which type of file (i.e. *prefix*`.grok` or `.mprops`) that they are placed in. It is very important that the user understand this behaviour, and the scope of each instruction will be clearly indicated as they are introduced and discussed below.

## Time varying pm zones from ascii file

`Scope: .grok`

1. **time(i), filename(i)...end** Time [T] and filename (up to 256 characters) list.

2. **tecplot_output** Logical value (T/F), which if true (T) writes the time-varying zones to the Tecplot ASCII file *prefix*`o.time_varying_pm_zones.dat`.

This command allows for time-varying porous medium zones by assigning zones from the time-file table. The input files consist of a list of element number and zone number pairs, one entry per line. For example, if a file is used to assign elements 1–5 zone number 1, then its format would be

```
1  1
2  1
3  1
4  1
5  1
```

At time **time(i)** the file **filename(i)** is applied and maintained until time **time(i + 1)**. The last file entered in the list will be applied until the end of the simulation. Note that porous medium material properties in porous medium properties files (`.mprops`) will not change over time, rather it is the spatial distribution of the porous medium zones that changes over time.

•••

## K isotropic

`Scope: .grok .mprops`

1. **Kval** Hydraulic conductivity [L $T^{-1}$].

Assign an isotropic hydraulic conductivity (i.e., $K_{xx} = K_{yy} = K_{zz} =$ **Kval**).

•••

## K anisotropic

`Scope: .grok .mprops`

1. **Kxx, Kyy, Kzz** Hydraulic conductivities [L $T^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively.

Assigns anisotropic hydraulic conductivities.

•••

## K anisotropic by ratio

`Scope: .grok .mprops`

1. **Kval, Kratio** Hydraulic conductivity [L $T^{-1}$] in the $x$- and $y$-directions and a ratio to compute the hydraulic conductivity in the $z$-direction.

Assigns anisotropic hydraulic conductivities such that $K_{xx} = K_{yy} =$ **Kval** and $K_{zz} =$ **Kratio** · **Kval**.

●●●

## K tensor

Scope: `.grok .mprops`

1. **Kxx, Kyy, Kzz** Diagonal terms of the hydraulic conductivity tensor: $K_{xx}, K_{yy}, K_{zz}$ [L T$^{-1}$].
2. **Kxy, Kxz, Kyz** Off-diagonal terms of the hydraulic conductivity tensor: $K_{xy}, K_{xz}, K_{yz}$ [L T$^{-1}$].

Assign hydraulic conductivities that include the off-diagonal terms. Note that this option will only work if **HydroGeoSphere** is running in finite element mode and so this switch will automatically be set.

●●●

## Specific storage

Scope: `.grok .mprops`

1. **val** Specific storage [L$^{-1}$], $S_s$ in Equation 2.10.

●●●

## Porosity

Scope: `.grok .mprops`

1. **val** Porosity [L$^3$ L$^{-3}$], $\theta_s$ in Equation 2.1.

●●●

## Poisson ratio

Scope: `.grok .mprops`

1. **val** Poisson's Ratio [-], $\nu^*$ in Equation 2.28b.

●●●

## Element K isotropic

Scope: `.grok`

1. **Kval** Hydraulic conductivity [L $T^{-1}$].

Chosen elements are assigned isotropic hydraulic conductivities (i.e., $K_{xx} = K_{yy} = K_{zz} =$ **Kval**).

●●●

## Element K anisotropic

Scope: .grok

1. **Kxx, Kyy, Kzz** Hydraulic conductivities [L $T^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively.

Chosen elements are assigned anisotropic hydraulic conductivities.

●●●

## Read elemental K from file

Scope: .grok

1. **filename** Name of the file which contains the variable K [L $T^{-1}$] data.

This file should contain one of the following input data formats:

1. **element_number, Kxx** Element number, isotropic hydraulic conductivity [L $T^{-1}$].
2. **element_number, Kxx, Kyy, Kzz** Element number, hydraulic conductivities [L $T^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively.
3. **element_number, Kxx, Kyy, Kzz, Kxy, Kyz, Kzx** Element number, hydraulic conductivity matrix components [L $T^{-1}$].

All elements are assigned a variable K from the file. For example, if there are four elements, with $K_{xx} = K_{yy} = 5$ m $d^{-1}$ and $K_{zz} = 2$ m $d^{-1}$, the file would contain:

```
1   5.0   5.0   2.0
2   5.0   5.0   2.0
3   5.0   5.0   2.0
4   5.0   5.0   2.0
```

The user can supply variable values for any number of elements in the input file. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the binary output file *prefix*`o.ElemK_pm.0001`.

●●●

## Read elemental K from binary file

`Scope: .grok`

1. **filename** Name of the binary file that contains the hydraulic conductivity values.

The file must be formatted as follows:

1. **bom** Byte order mark, the number 32,767 (2-byte signed integer).
2. **nelems** Number of elements being assigned values (4-byte signed integer).
3. **nprops** Number of properties per element (1, 3, or 6) (4-byte signed integer).
4. **elemIds(i), i=1,nelems** Element numbers (4-byte signed integers).
5. The property value block consists of **nelems** $\times$ **nprops** 8-byte real values. Its format depends on the value of **nprops**:
   (a) **nprops** = 1: **Kxx(i), i=1,nelems** Isotropic hydraulic conductivity [L $T^{-1}$].
   (b) **nprops** = 3: **Kxx(i), Kyy(i), Kzz(i), i=1,nelems** Hydraulic conductivities [L $T^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively.
   (c) **nprops** = 6: **Kxx(i), Kyy(i), Kzz(i), Kxy(i), Kyz(i), Kzx(i), i=1,nelems** Hydraulic conductivity [L $T^{-1}$] matrix components.

Assigns hydraulic conductivity values from the input file to the specified set of elements. Any previously set zoned values or user specified element-variable values will be honoured. Note that if the input file is being generated by Fortran, then it must be opened with the access specifier `access='stream'` to avoid the 4-byte record headers that are added when writing with sequential access. Conversion to the correct endianness is handled automatically when reading the file.

●●●

## Read elemental porosity from file

`Scope: .grok`

1. **filename** Name of the file which contains the variable porosity [-] data.

The input file should contain the following data:

1. **element_number, por** Element number, porosity [-].

Data is read from the file until end-of-file is reached. The user can supply variable porosity values for any number of elements in the input file. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the binary output file *prefix*`o.ElemPor_pm.0001`.

● ● ●

## Read elemental porosity from binary file

`Scope: .grok`

1. **filename** Name of the binary file that contains the porosity [-] values.

The file must be formatted as follows:

1. **bom** Byte order mark, the number 32,767 (2-byte signed integer).
2. **nelems** Number of elements being assigned values (4-byte signed integer).
3. **elemIds(i), i=1,nelems** Element numbers (4-byte signed integer).
4. **por(i), i=1,nelems** Porosity [-] values (8-byte real).

Assigns porosity values from the input file to the specified set of elements. Any previously set zoned values or user specified element-variable values will be honoured. Note that if the input file is being generated by Fortran, then it must be opened with the access specifier `access='stream'` to avoid the 4-byte record headers that are added when writing with sequential access. Conversion to the correct endianness is handled automatically when reading the file.

● ● ●

## Read elemental specific storage from file

`Scope: .grok`

1. **filename** Name of the file which contains the variable specific storage data.

The input file should contain the following data:

1. **element_number, stor** Element number, specific storage [$L^{-1}$], $S_s$ in Equation 2.10.

Data is read from the file until end-of-file is reached. The user can supply variable specific storage values for any number of elements in the input file. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the binary output file *prefix*`o.ElemStor_pm.0001`.

•••

## Read elemental specific storage from binary file

`Scope: .grok`

1. **filename** Name of the binary file that contains the specific storage values.

The file must be formatted as follows:

1. **bom** Byte order mark, the number 32,767 (2-byte signed integer).
2. **nelems** Number of elements being assigned values (4-byte signed integer).
3. **elemIds(i), i=1,nelems** Element numbers (4-byte signed integer).
4. **stor(i), i=1,nelems** Specific storage [$\mathrm{L}^{-1}$] values (8-byte real).

Assigns specific storage values from the input file to the specified set of elements. Any previously set zoned values or user specified element-variable values will be honoured. Note that if the input file is being generated by Fortran, then it must be opened with the access specifier `access='stream'` to avoid the 4-byte record headers that are added when writing with sequential access. Conversion to the correct endianness is handled automatically when reading the file.

•••

## Map property from raster for chosen elements

`Scope: .grok`

1. **filename, (band)** Name of raster file that contains the property values and optional raster band id (default value 1). Filenames with spaces must be enclosed by double quotes (“”).
2. **propid** Name of property to be updated. It can be one of the following:

| | |
|---|---|
| k isotropic | Updates $K_{xx} = K_{yy} = K_{zz}$ |
| kxx | Updates $K_{xx}$ |
| kyy | Updates $K_{yy}$ |
| kzz | Updates $K_{zz}$ |
| porosity | Updates $\theta_s$ |
| tortuosity | Updates $\tau$ |
| specific storage | Updates $S_s$ |

3. **interpolate** Logical value (T/F), which if true, causes property values to be interpolated from the raster via bilinear interpolation. Otherwise, property values correspond directly to raster cell values.

For each chosen element, the value of the property specified by **propid** is updated from the specified raster band cell that contains the 2-D centroid of that element.

•••

## Write element K

`Scope: .grok`

1. **filename** Name of the file in which to write the hydraulic conductivity [L $T^{-1}$] information, at most 80 characters.

Writes an ASCII file of elemental hydraulic conductivity values. The file is formatted as follows:

```
1 Kxx_1 Kyy_1 Kzz_1
2 Kxx_2 Kyy_2 Kzz_2
:   :     :     :
:   :     :     :
n Kxx_n Kyy_n Kzz_n
```

Here `Kxx_i`, `Kyy_i`, and `Kzz_i` are the hydraulic conductivity values in the three principal directions for the $i$th mesh element, where the index $i$ ranges from one to the number of mesh elements $n$.

•••

## Write element K at z

`Scope: .grok`

1. **zfix** $z$-coordinate [L] for choosing which element to write.
2. **rtol** Distance [L] from **zfix** for test.
3. **filenm** Name of the file in which to write the hydraulic conductivity [L $T^{-1}$] information.

This command is identical to the command Write element K except that information if only written for elements whose centroid is within a distance of **rtol** of the $z$-coordinate **zfix** are written to the file.

•••

## Get average K

`Scope: .grok`

For the group of currently chosen elements, this instruction computes the average hydraulic conductivity [L $T^{-1}$] and writes it to the *prefix*`o.lst` file. This is useful for example, when a random conductivity field has been generated and the user would like to know the average hydraulic conductivity of a region of the domain.

●●●

## Soil frost K

`Scope: .grok`

1. **npanel** Number of panels.
2. **ton_val(i), toff_val(i), Kval(i), i=1,npanel** Time on [T], time off [T], and soil frost hydraulic conductivity [L $T^{-1}$] for each panel.

Porous media hydraulic conductivity values are modified according to the input time on/off table for the current set of chosen elements. The hydraulic conductivity for each chosen element $j$ is updated according to

$$K_{\text{new},j} = \begin{cases} K_i, & t_{\text{on}}(i) < t \leq t_{\text{off}}(i) \text{ for some } i \in \{1, \ldots, n\} \\ K_{0,j}, & \text{otherwise} \end{cases},$$

where $n$ is the number of panels, $K_{0,j}$ is the initial hydraulic conductivity at $j$, and $K_i$ is the specified value.

●●●

## Soil frost K by ratio

`Scope: .grok`

1. **npanel** Number of panels.
2. **ton_val(i), toff_val(i), Kratio(i), i=1,npanel** Time on [T], time off [T], and the ratio [-] between soil frost hydraulic conductivity[L $T^{-1}$] and the original soil conductivity [L $T^{-1}$] for each panel.

Porous media hydraulic conductivity values are modified according to the input time on/off table for the current set of chosen elements. The hydraulic conductivity for each chosen element $j$ is updated according to

$$K_{\text{new},j} = \begin{cases} K_{0,j} \cdot K_i, & t_{\text{on}}(i) < t \leq t_{\text{off}}(i) \text{ for some } i \in \{1, \ldots, n\} \\ K_{0,j}, & \text{otherwise} \end{cases},$$

where $n$ is the number of panels, $K_{0,j}$ is the initial hydraulic conductivity at $j$, and $K_i$ is the specified ratio.

•••

## Time dependent K for chosen elements

`Scope: .grok`

1. **npanel** Number of panels.
2. **ton_val(i), Kxx(i), Kyy(i), Kzz(i), i=1,npanel** Time on [T] and hydraulic conductivities [L T$^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively, for each panel. The time values must satisfy the following set of inequalities:

   $$t_{\text{on}}(1) < t_{\text{on}}(2) < \cdots < t_{\text{on}}(n),$$

   where $n$ is the number of panels.

This command modifies the hydraulic conductivity for chosen elements such that

$$K_{\text{new}} = \begin{cases} K_i, & t_{\text{on}}(i) < t \leq t_{\text{on}}(i+1) \text{ for some } i \in \{1, \ldots, n-1\} \\ K_n, & t > t_{\text{on}}(n) \end{cases}$$

Note that if $t \leq t_{\text{on}}(1)$ then the hydraulic conductivity is unchanged. In the event that this command is issued more than once with common elements, for each common element the multiple time series are merged. In the unlikely event of identical **ton_val** values between multiple time series for a single element, the hydraulic conductivity that corresponds to the most recent issuing of the command will be honoured.

•••

## Time dependent variable K for chosen elements

`Scope: .grok`

1. **npanel** Number of panels.
2. **ton_val(i), filename(i), i=1,npanel** Time on [T] and the filename where hydraulic conductivity values are listed for each panel. Each line of an input file specifies the hydraulic conductivities [L T$^{-1}$] in the $x$-, $y$-, and $z$-directions (i.e., $K_{xx}$ $K_{yy}$ $K_{zz}$) for each chosen element. The time values must satisfy the following set of inequalities:

   $$t_{\text{on}}(1) < t_{\text{on}}(2) < \cdots < t_{\text{on}}(n),$$

   where $n$ is the number of panels.

This command modifies the hydraulic conductivity for chosen elements such that

$$K_{\text{new},j} = \begin{cases} K_{ij}, & t_{\text{on}}(i) < t \leq t_{\text{on}}(i+1) \text{ for some } i \in \{1, \ldots, n-1\} \\ K_{nj}, & t > t_{\text{on}}(n) \end{cases} ,$$

where $j$ is the spatial index. Note that if $t \leq t_{\text{on}}(1)$ then the hydraulic conductivity is unchanged. In the event that this command is issued more than once with common elements, for each common element the multiple time series are merged. In the unlikely event of identical **ton_val** values between multiple time series for a single element, the hydraulic conductivity that corresponds to the most recent issuing of the command will be honoured.

• • •

## K reduction by impedance factor for chosen elements

`Scope: .grok`

1. **impedance** Impedance factor ($> 0$) [-].
2. **Kmin** Minimum reduced hydraulic conductivity [L $\text{T}^{-1}$].
3. **npanel** Number of time panels in the time on/off table.
4. **ton_val(i), toff_val(i), i=1,npanel** Time on [T] and time off [T] for each panel. The time values must satisfy the following set of inequalities:

$$t_{\text{on}}(1) < t_{\text{off}}(1) < t_{\text{on}}(2) < t_{\text{off}}(2) < \cdots < t_{\text{on}}(n) < t_{\text{off}}(n),$$

   where $n$ is the number of panels.

This command, which applies to the porous media domain, modifies hydraulic conductivities [L $\text{T}^{-1}$] in the $x$-, $y$-, and $z$-directions for the current set of chosen elements. In particular, at the $i$th time-on panel, for each chosen element

$$K_{\text{new},j} = \min\left(\max\left(K_j \cdot 10^{-\Omega \cdot S_w(t=t_{\text{on}}(i))}, K_{min}\right), K_j\right) \quad \text{for} \quad t_{\text{on}}(i) < t \leq t_{\text{off}}(i),$$

where $\Omega$ is the impedance factor, which represents the maximum K-reduction in log-scale, $K_{min}$ is the minimum reduced hydraulic conductivity over all coordinate directions, and $j$ is the spatial index. Note that if $t < t_{\text{on}}(1)$ then the hydraulic conductivity is unchanged. For example:

```
k reduction by impedance factor for chosen elements
1.4 ! impedance factor
0.0 ! Kmin
8   ! number of time panels
! time on    time off
        0.0    7689600.0
 28339200.0   37584000.0
 59184000.0   70675200.0
 90547200.0  100396800.0
120960000.0  134697600.0
152841600.0  166838400.0
```

```
184723200.0  195004800.0
217123200.0  220665600.0
```

•••

## Time varying K by impedance factor via temperature raster

Scope: `.grok`

1. **impedance** Impedance factor (> 0) [-].
2. **Kmin** Minimum reduced hydraulic conductivity [L $T^{-1}$].
3. **freezing_temp** Freezing temperature [°C].
4. **interpolate** Logical value (T/F), which if true, causes temperature values to be linearly interpolated over a time panel. Otherwise, temperature values are read from the raster at the left endpoint of a time panel.
5. **time(i), filename(i), (band(i))...end** Time [T], temperature [°C] raster filename, and optional raster band id (default value 1) list. All filenames with spaces must be enclosed by double quotes (“”) and times must be given in strictly increasing order.

This command, which applies to the porous media domain, uses the table of temperature raster files to modify hydraulic conductivities [L $T^{-1}$] in the $x$-, $y$-, and $z$-directions for a set of chosen (near-surface) elements when the temperature drops below the freezing temperature. In particular, over the $i$th time panel for each chosen element

$$K_{\text{new},j} = \begin{cases} \min\left(\max\left(K_j \cdot 10^{-\Omega \cdot S_w(t=t_i)}, K_{min}\right), K_j\right) & T(t) < T_f \\ K_j, & \text{otherwise} \end{cases} \quad \forall t \in (t_i, t_{i+1}],$$

where $\Omega$ is the impedance factor, which represents the maximum K-reduction in log-scale during the freezing season, $K_{min}$ is the minimum reduced hydraulic conductivity over all coordinate directions, $T_f$ is the freezing temperature, $T(t)$ is the temperature over the time panel at time $t$, and $j$ is the spatial index. Note that if $t < t_1$ (the first time in the table) then the hydraulic conductivity is unchanged. In the event of common elements among multiple command invocations, the reduction factor corresponding to the most recent issuing of the command will be used. For example:

```
time varying k by impedance factor via temperature raster
7.0 ! impedance factor
0.0 ! Kmin
0.0 ! freezing temperature
T   ! true to use interpolation
! times       raster files
       0.0    temperature1.asc
```

```
  2592000.0    temperature2.asc
  5184000.0    temperature3.asc
  7776000.0    temperature4.asc
 10368000.0    temperature5.asc
 12960000.0    temperature6.asc
 end
```

●●●

## Permafrost formation from file

Scope: `.grok`

1. **filename** Filename of the ASCII file containing permafrost formation locations and times.

The input file consists of blocks of the form:

```
element id, number of times (n)
t1, T/F
t2, T/F
:    :
:    :
tn, T/F
```

Each block defines a time on/off table for a given element in the mesh. The time values should be listed in increasing order, i.e.,

$$t_1 < t_2 < \cdots < t_n.$$

This command can be used in conjunction with the following commands: Permafrost k, Permafrost porosity, Permafrost effective diffusion coefficient. When used as part of a transport simulation this command should be placed after the Solute...End command block that defines solute properties; otherwise, **grok** will terminate with an error message.

For each input block, let $j$ denote the element id, $n$ the number of times, $P_j$ the specified value of a property to be updated, $P_{0,j}$ the initial value of that property, and $b_i$ the numeric value associated with the boolean value at time $t_i$ in the table under the mapping

$$\begin{aligned} \mathrm{T} &\mapsto 1 \\ \mathrm{F} &\mapsto 0 \end{aligned}$$

Then the updated property value $P_{\text{new},j}$ at simulation time $t \in [t_i, t_{i+1}]$ is computed via linear interpolation between the points $(t_i, P_j^L)$ and $(t_{i+1}, P_j^R)$ with

$$\begin{aligned} P_j^L &= b_i P_j + (1 - b_i) P_{0,j}, \\ P_j^R &= b_{i+1} P_j + (1 - b_{i+1}) P_{0,j}. \end{aligned}$$

Note that if a model defines the fracture domain, then at fracture faces adjacent to a permafrost element, the minimum porous medium hydraulic conductivity value over all coordinate directions will be assigned as the fracture hydraulic conductivity.

● ● ●

## Permafrost k

`Scope: .grok`

1. **Kxx, Kyy, Kzz** Permafrost hydraulic conductivities [L $T^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively.

Assigns anisotropic hydraulic conductivities of permafrost. The assigned values change with respect to the times defined by the command Permafrost formation from file.

● ● ●

## Permafrost porosity

`Scope: .grok`

1. **porosity** Permafrost porosity [-].

Assigns the porosity of permafrost. The assigned values change with respect to the times defined by the command Permafrost formation from file.

● ● ●

## Permafrost effective diffusion coefficient

`Scope: .grok`

1. **diff_coeff** Permafrost effective diffusion coefficient [$L^2$ $T^{-1}$].

Assigns the effective diffusion coefficient of permafrost. The assigned values change with respect to the times defined by the command Permafrost formation from file.

● ● ●

#### 2.8.2.2 Discrete Fractures

Unless you modify the default values, all discrete fracture zones (and 2-D fracture elements) in the domain will be assigned the default saturated properties which are listed in Table 2.7. These values are representative of a 100 micron fracture.

Table 2.7: Default values for fractured media saturated flow properties.

| Parameter | Value | Unit |
|---|---|---|
| Name | Default 100 micron fracture | |
| Aperture $w_f$ | $1.0 \times 10^{-4}$ | m |
| Conductivity (computed) $K_f$ | $(\rho g \, w_f^2)/(12\mu)$ | m s$^{-1}$ |
| Specific storage (computed) $S_{sf}$ | $\rho g \alpha_w$ | m$^{-1}$ |
| Coupling length | $1.0 \times 10^{-4}$ | m |
| Coupling conductivity | 0.0 | m s$^{-1}$ |
| Unsaturated flow relation type | Pseudo-fracture | |

You can use the general methods and instructions outlined in Section 2.8.1 to modify the default distribution of saturated fractured media properties.

As was the case for the instructions which modify porous medium properties, these instructions also have a scope of operation, the only difference being that they would appear in the `.fprops` file instead of the `.mprops` file.

## Aperture

`Scope: .grok .fprops`

1. **val** Fracture aperture [L], $w_f$ in Equation 2.11.

•••

## High-K plane

`Scope:  .fprops`

1. **valx** Thickness [L] for the high-conductivity plane.
2. **valx** Hydraulic conductivity [L T$^{-1}$] for the high-conductivity plane.

Treats the fracture material as a high-conductivity plane where the hydraulic conductivity and thickness are given by the user.

•••

## Fracture zone porosity

`Scope:  .fprops`

1. **valx** Porosity [-] for the high-conductivity plane.

In the case that fracture aperture is given, the porosity is assumed to be unity and this command has no effect.

●●●

## Specific storage

`Scope: .grok .fprops`

1. **val** Specific storage [$L^{-1}$], $S_{sf}$ in Equation 2.14.

●●●

## Coupling length

`Scope: .grok .fprops`

1. **val** Coupling length [L].

●●●

## Coupling hydraulic conductivity

`Scope: .grok .fprops`

1. **val** Coupling hydraulic conductivity [$L\ T^{-1}$].

●●●

## Impermeable matrix

`Scope: .grok`

This command causes the matrix to be considered impermeable and so flow and transport will only be computed for the fractures. This overrides any values defined in the *prefix*`.grok` or `.fprops` file so you do not have to alter them.

●●●

## Generate individual fracture and read parameters from file

`Scope: .grok`

1. **filename** Name of file that contains the fracture information.
2. **tecplot_output** Logical value (T/F), which if true, causes the fracture parameters for all fracture elements to be written a Tecplot formatted ASCII output file. If true, then read the following:
   (a) **tecplot_filename** Filename of Tecplot formatted ASCII output file. Note that it should have the file extension `.dat`.

This command assigns individual fracture information to a fracture domain mapped onto either a triangular prism or hexahedral block element mesh. The format of the input file is as follows:

1. **header** Header line (can be left blank).
2. **nelems** Number of fracture elements.
3. **header** Header line (can be left blank).
4. **id(i), nd1(i), nd2(i), nd3(i), thk(i), cond(i), por(i), stor(i), i=1,nelems** Element id, node ids (triangle), thickness [L], hydraulic conductivity [L $T^{-1}$], porosity [-], and specific storage [$L^{-1}$] of a mapped fracture element.

For example:

```
number of fracture elements
2563
FracElemID ENode1 ENode2 ENode3 Thick HydCond Porosity SpecStorage
1   12832   16407   16343   1.00E-04    1.00E-04    0.4 1.00E-03
2   12832   16471   16407   1.00E-04    1.00E-04    0.4 1.00E-03
3   13087   16662   16598   1.00E-04    1.00E-04    0.4 1.00E-03
4   13087   16726   16662   1.00E-04    1.00E-04    0.4 1.00E-03
:      :       :       :        :           :          :      :
```

Note that fracture element ids must be listed in increasing order from one to the number of fracture elements.

•••

## Map fractures from mofrac vtk file

Scope: .grok

1. **vtk_filename** Filename of the MoFrac VTK input file.
2. **mapped_filename** Filename of the mapped fracture output file.
3. **num_fracs** Maximum number of fractures in the MoFrac VTK input file.
4. **tecplot_output** Logical value (T/F), which if true, causes the MoFrac VTK input file to be written to a Tecplot formatted ASCII output file. If true, then read the following:
   (a) **tecplot_filename** Filename of the Tecplot formatted ASCII output file. Note that it should have the file extension `.dat`.

This command maps fractures that are triangulated and stored in the MoFrac VTK format. Fractures are mapped as triangles to match the input fracture geometry and the mapping is applicable to both 3-D hexahedral and prism element mesh types. An example of using this command after a fracture domain has been defined is as follows.

```
use domain type
fracture

properties
../prop/fracture.fprops

map fractures from mofrac vtk file
fracture.vtk
fracture_mapped.txt
10
T
fracture_tecplot.dat
```

This command generates the following output files:

```
MappedFrac_VTKFaceid.dat
Mapped_FracElem_Param.dat
VTK_Frac_To_Frac_ElemID.dat
Mapped_Fracture_FracIn.tec.dat
```

● ● ●

## Map fractures from tecplot file

`Scope: .grok`

1. **tecplot_frac_file** Name of the Tecplot file which contains the fracture triangle information.
2. **label** Name of the fracture material whose properties are to be read and assigned.

A new fracture zone is created based on triangle data given in the Tecplot file **tecplot_frac_file**. Fracture zone properties are assigned from the material **label**.

The Tecplot file must contain one or more Tecplot ZONE statements. Each Tecplot ZONE statement must contain, or be followed by a statement containing the string `N=` followed by the number of vertices and then by the string `E=` followed by the number of triangles in the Tecplot ZONE.

Comments beginning with the character # are allowed.

The Tecplot variables represent the *xyz*-coordinates of the vertices and must be given in three-column, point format.

The fracture elements are defined so that they conform closely to the triangulated surface given in the Tecplot file, and can include diagonal (i.e., inclined) faces. For example, the fracture elements shown in Figure 2.12 were generated from the following Tecplot file:

```
VARIABLES="X", "Y", "Z"
ZONE T="Fractures_primary"
N=5, E=3, DATAPACKING=POINT, ZONETYPE=FETRIANGLE
110. -1. -5.
51.  102.   -3.
-3.  49.  99.
0. 100. 100
0. 0. 0.
1 3 5
1 2 3
2 3 4
```

•••

#### 2.8.2.3 Dual Continuum

Unless you modify the default values, all dual-continuum zones (and elements) in the domain will be assigned the default properties which are listed in Table 2.8. These values are representative of a sand.

Table 2.8: Default values for dual-continuum saturated flow properties.

| Parameter | Value | Unit |
|---|---|---|
| Name | Default Sand | |
| Hydraulic conductivity terms: | | |
| $K_{xx}$ | $7.438\times10^{-5}$ | m s$^{-1}$ |
| $K_{yy}$ | $7.438\times10^{-5}$ | m s$^{-1}$ |
| $K_{zz}$ | $7.438\times10^{-5}$ | m s$^{-1}$ |
| Specific storage $S_{sd}$ | $1.0\times10^{-4}$ | m$^{-1}$ |
| Porosity $\theta_{sd}$ | 0.375 | |
| Volume fraction of porous medium $w_d$ | 0.01 | |
| Unsaturated flow relation type | Pseudo-soil | |

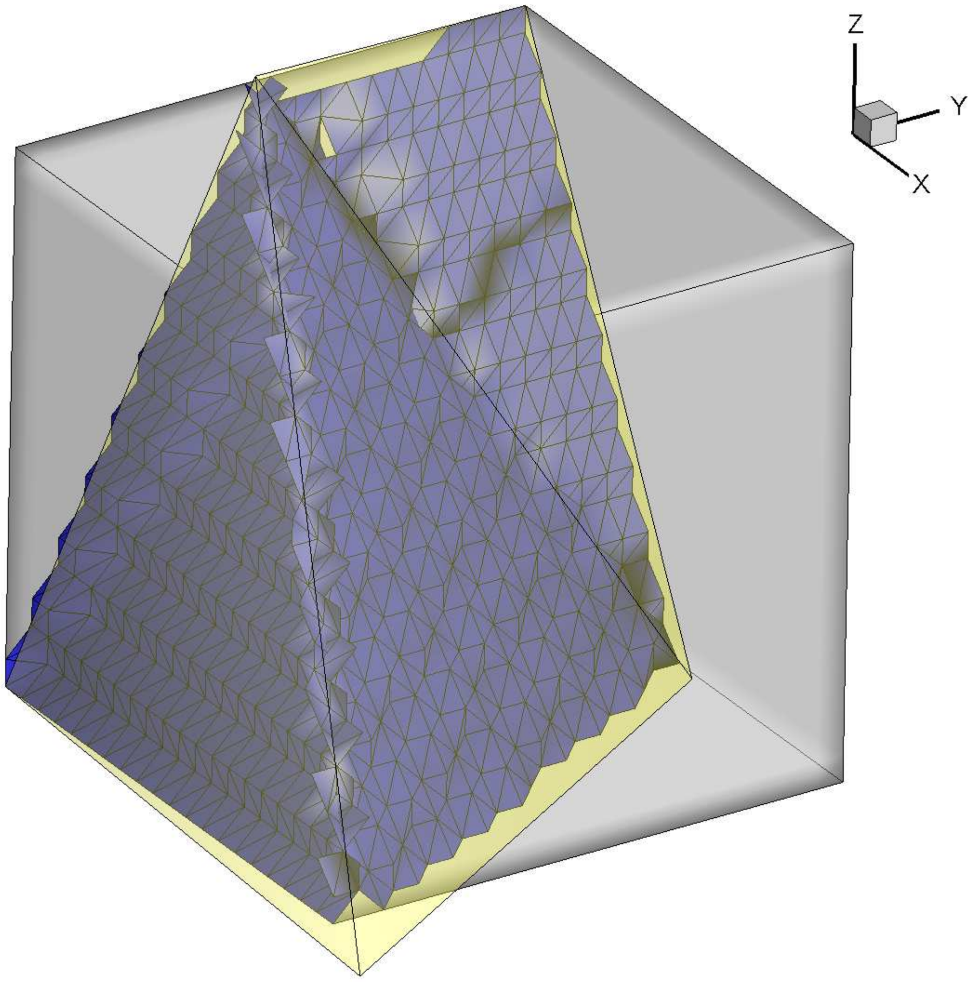


Figure 2.12: Example of fracture generation showing 3-D porous medium domain (grey cube), Tecplot triangles (yellow triangles), and the resulting fracture elements (blue triangles).

Note that the default state of the hydraulic conductivity tensor ($\mathbf{K_d}$ in Equation 2.17) is that it is isotropic. It should also be noted that for dual continua, off-diagonal terms are not considered.

You can use the general methods and instructions outlined in Section 2.8.1 to modify the default distribution of saturated dual-continuum properties.

As was the case for the instructions which modify porous medium properties, these instructions also have a scope of operation, the only difference being that they would appear in the `.dprops` file instead of the `.mprops` file.

## K isotropic

`Scope: .grok .dprops`

1. **Kval** Hydraulic conductivity [L T$^{-1}$].

Assign an isotropic hydraulic conductivity (i.e., $K_{xxd} = K_{yyd} = K_{zzd} =$ **Kval**).

• • •

## K anisotropic

`Scope: .grok .dprops`

1. **Kxx, Kyy, Kzz** Hydraulic conductivities [L T$^{-1}$] in the $x$-, $y$-, and $z$-directions, respectively.

Assigns anisotropic hydraulic conductivities.

• • •

## K anisotropic by ratio

`Scope: .grok .dprops`

1. **Kval, Kratio** Hydraulic conductivity [L T$^{-1}$] in the $x$- and $y$-directions and a ratio to compute the hydraulic conductivity in the $z$-direction.

Assigns anisotropic hydraulic conductivities such that $K_{xxd} = K_{yyd} =$ **Kval** and $K_{zzd} =$ **Kratio** $\cdot$ **Kval**.

• • •

## Specific storage

`Scope: .grok .dprops`

1. **val** Specific storage [L$^{-1}$], $S_{sd}$, but defined in a similar way to $S_s$ in Equation 2.10.

• • •

## Porosity

`Scope: .grok .dprops`

1. **val** Porosity [L$^3$ L$^{-3}$], $\theta_{sd}$ in Equation 2.16.

• • •

## Volume fraction dual medium

`Scope:  .dprops`

1. **val** Volume fraction [$L^3$ $L^{-3}$], $w_d$ in Equation 2.16.

The volume fractions of the dual medium and porous medium always add up to 1.0.

●●●

## First-order fluid exchange coefficient

`Scope: .dprops`

1. **val** Exchange coefficient [$L^{-2}$], $\alpha^*_{wd}$ in Equation 2.79.

●●●

## Interface k

`Scope: .dprops`

1. **val** Interface hydraulic conductivity [L $T^{-1}$], $K_a$ in Equation 2.78.

●●●

## Convert pm k to macropore k

`Scope: .dprops`

1. **val** Porous medium background hydraulic conductivity $K_{bkgrd}$ [L $T^{-1}$].

We can express the bulk hydraulic conductivity of a dual-continuum $K_{bulk}$ as the sum of the porous media $K_{bkgrd}$ and fracture $K_d$ components:

$$K_{bulk} = K_{bkgrd}\,(1 - w_d) + K_d\,w_d \tag{2.6}$$

where $w_d$ is the volume fraction [$L^3$ $L^{-3}$] in Equation 2.16.

If we assume that the observed (porous medium) hydraulic conductivity is equal to $K_{bulk}$ and supply an educated guess for $K_{bkgrd}$, then we can rearrange the equation and calculate $K_d$ as

$$K_d = \frac{K_{bulk} - K_{bkgrd}\,(1 - w_d)}{w_d} \tag{2.7}$$

For all elements in the currently chosen dual zones, the porous medium hydraulic conductivity is replaced by $K_{bkgrd}$ and the fracture hydraulic conductivity $K_d$ is set equal to the calculated value.

●●●

#### 2.8.2.4 Wells

Well domains, as outlined in Section 2.3.2.2 of the Theory Manual and in Equation 2.63, can be set up using the following instructions. The variable **radius** corresponds to $r_s$, while the pumping rates $Q_w$, use the boundary condition format described in Section 2.7. Unless the user modifies the default values, all well zones in the domain will be assigned the default properties. The default values, Table 2.9, can be and are recommend to be modified to fit the model application.

Table 2.9: Default values for well properties.

| Parameter | Value | Unit |
|---|---|---|
| Screen radius | $1.0\times10^{-5}$ | m |
| Coupling conductivity | $7.438\times10^{-5}$ | m s$^{-1}$ |
| Coupling length | $1.0\times10^{-4}$ | m |
| Well type | Hagen–Poiseuille | |
| Manning's friction coefficient | 0.0548 | m$^{-1/3}$ s |
| Hazen–Williams coefficient | 130.0 | m$^{0.37}$ s$^{-1}$ |

For example, the following set of instructions, with *prefix*`.wprops` file Figure 2.13, could be inserted into the *prefix*`.grok` file to produce a pumping well:

```
use domain type
well

properties file
prefix.wprops

clear chosen segments
choose segments polyline                    !Screen interval
2
100. 100. 0.
100. 100. 20.

new zone
1

clear chosen zones
choose zone number
1

read properties
well one

clear chosen nodes
```

```
choose node                                 !Bottom of pump intake
100. 100. 0.

create node set
well

boundary condition
    type
    flux nodal

    name
    well

    node set
    well

    time value table
    0.0  -80.0e-3                           !Pumping duration and rate
    end
end
```

The well domain example opens the property file *prefix*.`wprops`, shown in Figure 2.13, and creates a 20 m long screen located at $x = 100$ m and $y = 100$ m with a pump intake located at $z = 0$ m. The well one properties are read into **grok** and a steady pumping rate is set at $-80 \times 10^{-3}$ m$^3$ s$^{-1}$. If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

```
!------------------------------------------
well one

radius
0.05

coupling length
1.0e-4

coupling conductivity
0.0023

end
```

Figure 2.13: Example of a well domain property file *prefix*.`wprops`.

## Radius

`Scope: .grok .wprops`

1. **radius** Well Screen Radius [L].

The radius of the screen interval of the well.

• • •

## Infilled

`Scope: .wprops`

1. **infilled_name** Infilled well material name from porous medium `.mprops` file, up to 40 characters.

This command should be included in the property file for an infilled well. The default values are a sand as described in Table 2.6. Note that when running a transport simulation with infilled wells it is assumed that the effective diffusion coefficient of each solute is isotropic for the infill material. If the effective diffusion coefficient is defined as $\tau D_{free}$ (see Equation 2.122), then the tortuosity $\tau$ is required to be isotropic and must be defined on a zone-by-zone basis (i.e., it cannot vary on an elementwise basis).

• • •

## Hagen Poiseuille

`Scope: .grok .wprops`

This command should be included in the input for the Hagen–Poiseuille flow option.

• • •

## Hazen Williams

`Scope: .grok .wprops`

This command should be included in the input file for the Hazen–Williams flow option.

• • •

## Manning

`Scope: .grok .wprops`

This command should be included in the input file for the Manning flow option.

• • •

## Friction

Scope: `.grok .wprops`

1. **friction** Manning's friction [$L^{-1/3}$ T] for wells.

• • •

## Hazen Williams coefficient

Scope: `.grok .wprops`

1. **hw_coeff** Hazen–Williams coefficient [$L^{0.37}$ $T^{-1}$] for wells.

• • •

## Saturated wells

Scope: `.grok .wprops`
This instruction causes wells to remain fully saturated.

• • •

## Coupling conductivity

Scope: `.grok .wprops`

1. **cpl_cond** Coupling conductivity [L $T^{-1}$] for wells.

Coupling conductivity parameter only required for the dual node approach for wells.

• • •

## Coupling length

Scope: `.grok .wprops`

1. **cpl_length** Coupling length [L] for wells.

Coupling length parameter only required for the dual node approach for wells.

• • •

#### 2.8.2.5 Tile Drains

Tile drains, as outlined in Section 2.3.2.3 of the Theory Manual, can be set up using the following instructions. Currently, this option requires that the system be variably-saturated.

The default tile drain values, Table 2.10, can be and are recommend to be modified to fit the model application.

Table 2.10: Default values for tile drain properties.

| Parameter | Value | Unit |
|---|---|---|
| Screen radius | $1.0 \times 10^{-5}$ | m |
| Coupling conductivity | $7.438{\times}10^{-5}$ | m $s^{-1}$ |
| Coupling length | $1.0 \times 10^{-4}$ | m |
| Tile drain type | Manning | |
| Manning's friction coefficient | 0.0548 | $m^{-1/3}$ s |
| Hazen–Williams coefficient | 130.0 | $m^{0.37}$ $s^{-1}$ |

For example, the following set of instructions, with *prefix*.`tprops` file Figure 2.14, could be inserted into the *prefix*.`grok` file to produce a horizontal tile drain:

```
use domain type
tile

properties file
prefix.tprops

clear chosen segments
choose segments polyline                ! Screen interval
2
15. 1.   2.
15. 29. 2.

new zone
1

clear chosen zones
choose zone number
1

read properties
tile one

clear chosen nodes
choose node                             ! Tile drain intake
15. 15. 2.

create node set
tile
```

```
boundary condition
    type
    flux nodal

    name
    tile one

    node set
    tile

    time value table
    0.0  -2                            ! Pumping duration and rate
    end
end
```

The tile domain example opens the property file *prefix*.`tprops`, shown in Figure 2.14, and creates a 28 m long screen between (15, 1, 2) and (15, 29, 2) with a pump intake located at (15, 15, 2). The `tile one` properties are read into **grok** and a steady pumping rate is set at $-2$ m$^3$ s$^{-1}$.

```
!-------------------------------------------
tile one

radius
0.0108

! Coupling length and conductivity for dual node approach
coupling length
1.0e-4

coupling conductivity
0.0023

! Infilled tiled drain with properties from the mprops file
infilled
porous medium

end
```

Figure 2.14: Example of a tile drain domain property file *prefix*.`tprops`.

## Radius

`Scope: .grok .wprops`

1. **radius** Tile drain screen radius [L].

The radius of the screen interval of the tile drain.

●●●

## Infilled

`Scope: .tprops`

1. **infilled_name** Infilled tile material name from porous medium `.mprops` file, up to 40 characters.

This command should be included in the property file for an infilled tile. The default values are a sand as described in Table 2.6. Note that when running a transport simulation with infilled tiles it is assumed that the effective diffusion coefficient of each solute is isotropic for the infill material. If the effective diffusion coefficient is defined as $\tau D_{free}$ (see Equation 2.122), then the tortuosity $\tau$ is required to be isotropic and must be defined on a zone-by-zone basis (i.e., it cannot vary on an elementwise basis).

●●●

## Hagen Poiseuille

`Scope: .grok .tprops`

This command should be included in the input for the Hagen–Poiseuille flow option.

●●●

## Hazen Williams

`Scope: .grok .tprops`

This command should be included in the input file for the Hazen–Williams flow option.

●●●

## Manning

`Scope: .grok .tprops`

This command should be included in the input file for the Manning flow option.

●●●

## Friction

`Scope: .grok .tprops`

1. **friction** Manning's friction [$L^{-1/3}$ T] for tile drains.

●●●

---

### Hazen Williams coefficient

`Scope: .grok .tprops`

1. **hw_coeff** Hazen–Williams coefficient [$L^{0.37}$ $T^{-1}$] for tile drains.

●●●

---

### Coupling conductivity

`Scope: .grok .tprops`

1. **cpl_cond** Coupling conductivity [L $T^{-1}$] for tile drains.

Coupling conductivity parameter only required for the dual node approach for tile drains.

●●●

---

### Coupling length

`Scope: .grok .tprops`

1. **cpl_length** Coupling length [L] for tile drains.

Coupling length parameter only required for the dual node approach for tile drains.

●●●

#### 2.8.2.6 Channel Flow

The channel domain, which is outlined in Section 2.3.2.4 of the Theory Manual, can be set up using the following instructions. The default channel property values, Table 2.11, can be and are recommend to be modified to fit the model application.

For example, the following set of instructions, with *prefix*`.cprops` file Figure 2.15, could be inserted into the *prefix*`.grok` file to produce a channel:

```
use domain type
channel

properties file
```

Table 2.11: Default values for channel properties.

| Parameter | Value | Unit |
|---|---|---|
| Shape | Rectangle | |
| Width | 1.0 | m |
| Manning's friction coefficient | 0.0548 | $m^{-1/3}$ s |
| Rill storage height | 0.0 | m |
| Obstruction storage height | 0.0 | m |
| Stream bank height | 1.0 | m |
| Streambed thickness | 0.001 | m |
| Streambed conductivity | $7.438\times10^{-5}$ | m $s^{-1}$ |
| Incision depth | 0.0 | m |
| Weir discharge coefficient | 0.75 | |

```
prefixl.cprops

clear chosen segments
choose segments polyline
2
0.0  10.0  20.0
100.0  10.0  10.0

new zone
1

clear chosen zones
choose zone number
1

read properties
channel one

clear chosen nodes
choose nodes all
initial head surface elevation

clear chosen nodes
choose node
0.0  10.0  20.0

create node set
inlet_node

boundary condition
```

```
        type
        head

        name
        inlet_node_flux

        node set
        inlet_node

        time value table
        0.0  20.0001
        end
    end

    clear chosen nodes
    choose node
    100.0  10.0  10.0

    create node set
    outlet_node

    boundary condition
        type
        channel critical depth

        name
        cd_outlet_node

        node set
        outlet_node
    end

    clear chosen nodes
    choose nodes top block
    0.0 99.5
    9.0 11.
    0.0 20.
    create node set
    internal_node

    boundary condition
        type
        flux nodal

        name
        internal_flux

        node set
        internal_node

        time value table
        0.0  0.601818e-2
        end
    end
```

The channel domain example opens the property file *prefix*`.cprops`, shown in Figure 2.15, and creates a 1 m wide channel. The `channel one` properties are read into **grok**.

```
!-------------------------------------------
channel one

type rectangle
1.0

friction
0.01

rill storage height
0.001

obstruction storage height
0.0

streambed thickness
0.3

streambed conductivity
1.0e-4

bank height
1.0

end
```

Figure 2.15: Example of a channel domain property file *prefix*`.cprops`.

## Type rectangle

`Scope: .grok .cprops`

1. **width** Channel width [L].

Defines a rectangular stream channel cross section (see Figure 2.16).

●●●

## Type trapezoid

`Scope: .grok .cprops`

1. **width** Channel width [L] of the bottom of the channel.
2. **angle** Channel bank angle [deg] from the vertical line.

Defines a trapezoidal stream channel cross section. Equivalent to a rectangular cross section when the bank angle is zero degrees. Equivalent to a V-notch cross section when the channel bottom width is zero.

●●●

## Type circle

`Scope: .grok .cprops`

1. **radius** Channel radius [L].

Defines a semicircular stream channel cross section.

●●●

## Type general

`Scope: .grok .cprops`

1. **ntab** Number of water depth values in the table.
2. **depth(i), flow_area(i), wetted_perim(i), top_width(i), i=1,ntab** Water depth [L], area of flow [$L^2$], wetted perimeter [L], and top width [L].

Defines a general stream channel cross section via a table that specifies the area of flow, wetted perimeter, and top width for given water depth values.

●●●

## Friction

`Scope: .grok .cprops`

1. **friction** Manning friction [$L^{-1/3}$ T] for the channel.

Sets the Manning friction in the channel.

●●●

## Rill storage height

Scope: `.grok .cprops`

1. **rill_store** Channel rill storage height [L].

Sets the minimum water depth required for flow (see Section 2.2.2.2 of the Theory Manual).

●●●

## Obstruction storage height

Scope: `.grok .cprops`

1. **obstruction_store** Channel obstruction storage height [L].

Sets the obstruction storage height (see Section 2.2.2.2 of the Theory Manual).

●●●

The following commands, which are required only when using the dual node approach for channels, specify the parameters required to define the interaction between the channel domain and the overland flow and subsurface domains, as illustrated in Figure 2.16.

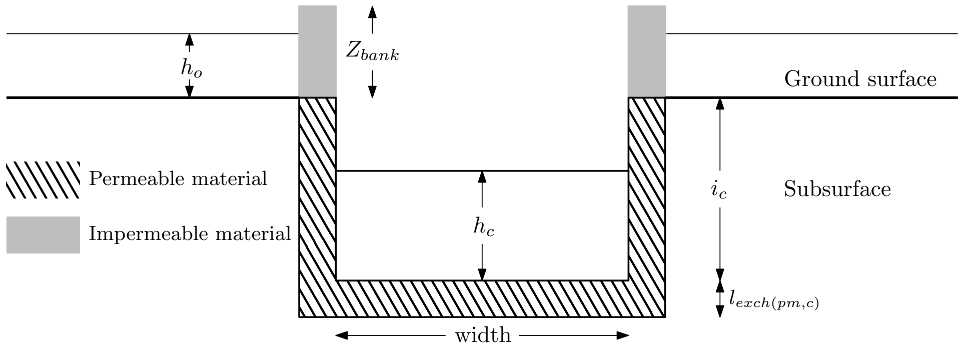


Figure 2.16: Rectangular 1-D channel cross section showing the stream bank height $Z_{bank}$, incision depth $i_c$, streambed thickness $l_{exch(pm,c)}$, overland head $h_o$, and channel head $h_c$.

## Streambed thickness

Scope: `.grok .cprops`

1. **thickness** Thickness of streambed [L].

This command, which is required only when using the dual node approach for channels, specifies the streambed thickness that defines the coupling between channel and subsurface. See variable $l_{exch(pm,c)}$ in Equation (2.75).

● ● ●

## Streambed conductivity

`Scope: .cprops`

1. **cond** Streambed hydraulic conductivity [L T$^{-1}$].

This command, which is required only when using the dual node approach for channels, specifies the streambed hydraulic conductivity. Together with streambed thickness this value defines the coupling conductance between channel and subsurface. See variable $K_{exch(pm,c)}$ in Equation (2.75).

● ● ●

## Time varying streambed conductivity

`Scope: .cprops`

1. **time(i), cond(i)...end** Time [T] and streambed hydraulic conductivity [L T$^{-1}$] list.

This command, which is required only when using the dual node approach for channels, specifies a time varying streambed hydraulic conductivity via a time-value table. See variable $K_{exch(pm,c)}$ in Equation (2.75).

● ● ●

## Bank height

`Scope: .cprops`

1. **height** Height of stream bank [L].

This command, which is required only when using the dual node approach for channels, specifies the stream bank height to define the interaction between the channel and overland flow domains. See variable $Z_{bank}$ in Equation (2.74).

● ● ●

## Incision depth

`Scope: .cprops`

1. **depth** Incision depth [L] for discharge to/from channels.

This command, which is required only when using the dual node approach for channels, specifies the incision depth that is used to define the interaction between the channel and overland flow domains, and between the channel and porous medium domains. See variable $i_c$ in Equations (2.74) and (2.75).

• • •

## Weir constant

`Scope: .cprops`

1. **weir_const** Weir discharge coefficient [-].

This command, which is required only when using the dual node approach for channels, specifies the weir discharge coefficient that is used to define the interaction between the channel and overland flow domains. See variable $C_d$ in Equation (2.74).

• • •

**Important:** When using the channel domain users should be aware of the limitations of open channel flow and take care not to create a physically unrealistic situation. The combination of the optional incision depth and bank height properties, in particular, can cause confusion. In all cases it is important to recognize that the location of channel domain nodes coincide with the location of surface domain nodes. In other words, regardless of channel incision depth, exchange between the 1-D channel and overland flow or porous medium domains occurs at the location of the node at the top of the model. With this in mind, it should be noted that while the incision depth may be greater than the thickness of the uppermost model layer and the channel may "conceptually" intersect several model layers, exchange between the channel and porous medium occurs only at the location of the node at the top of the model. Also note that a correction for the incision depth is applied automatically to initial depths defined for the 1-D channel domain, and therefore it is unrealistic to apply an initial head value which is less than the elevation of the node. See Section 2.3.2.4 of the Theory Manual for a description of how the bank height and incision depth affect exchange between the 1-D channel, overland flow, and porous medium domains.

### 2.8.3 Variably-Saturated Subsurface Flow

As discussed in Section 2.1 of the Theory Manual, one of the requirements for simulating variably-saturated flow is that we define the constitutive relationships that govern the relation between the pressure head, saturation and relative permeability. These relationships must be defined for the porous medium, discrete fractures and the dual continuum and, for each of these types of media, the choice is to use either pseudo-soil, functional or tabular relationships. Wells are a special case and are handled automatically by **HydroGeoSphere**.

Media specific instructions and default values for porous media, discrete fractures and dual continua are given in Sections 2.8.3.1, 2.8.3.2 and 2.8.3.3 respectively.

General instructions for modifying the default functional and tabular relationships are given in Sections 2.8.3.4 and 2.8.3.5 respectively.

#### 2.8.3.1 Porous Medium

By default, all porous media zones (and elements) in the domain will use a constitutive relationship based on the pseudo-soil relation developed by Huyakorn et al. (1994). Under this relationship, the medium is assigned a nodal saturation of zero above the water table and one below it. Relative permeability is applied to horizontal flow only and water travels vertically under saturated hydraulic conductivity conditions.

If you wish to use Van Genuchten or Brooks–Corey functions to describe the constitutive relationship you can do so using the instructions given in Section 2.8.3.4. Unless you modify them, the default values given in Table 2.12 will be used to define the functional relationships.

Table 2.12: Default values for functions defining the porous media constitutive relationships, for the Van Genuchten and Brooks–Corey models.

| Parameter | Value | Unit |
|---|---|---|
| **Common** | | |
| Residual water saturation, $S_{wr}$ | 0.18 | |
| Pore-connectivity, $l_p$ | 0.5 | |
| Beta (power index), $\beta$ | 6.0 | |
| **Van Genuchten** | | |
| Alpha (power index), $\alpha$ | 1.9 | $\mathrm{m}^{-1}$ |
| Gamma (Power index, computed), $\gamma$ | $1 - 1/\beta$ | |
| **Brooks–Corey** | | |
| Exponent (computed) | $2/\beta + l_p + 2$ | |
| Air entry pressure, $\psi_{ae}$ | 1.9 | m |
| Alpha (power index, computed), $\alpha$ | $-1/\psi_{ae}$ | $\mathrm{m}^{-1}$ |

If you wish to use tables to describe the constitutive relationships you can do so using the instructions given in Section 2.8.3.5. Unless you modify them, the default values of water saturation versus pressure head and saturation versus relative permeability listed in Table 2.13 will be used to define the tabular relationships.

## Relative permeability xy

`Scope: .grok .mprops`

When using functions or tables to define the constitutive relationships, this instruction causes **grok** to apply the relative permeability to horizontal flow only so that water travels vertically under saturated hydraulic conductivity conditions, similar to the behaviour of the pseudo-soil relation. This instruction should be applied to porous media, as discussed in Section 2.8.1.

●●●

Table 2.13: Default pressure-saturation and saturation-relative permeability tables for variably-saturated porous media.

| Pressure (m) $\psi$ | Saturation $S_w$ |
|---|---|
| $-10.0$ | 0.053 |
| 0.0 | 1.0 |

| Saturation $S_w$ | Relative Permeability $k_{rw}$ |
|---|---|
| 0.053 | 0.053 |
| 1.0 | 1.0 |

#### 2.8.3.2 Discrete Fractures

By default, all discrete fracture zones (and elements) in the domain will use a constitutive relationship based on the pseudo-soil relation developed by Huyakorn et al. (1994). Under this relationship, the medium is assigned a nodal saturation of zero above the water table and one below it. Also by default, the effective area available for flow across the fracture-matrix interface is maintained at its maximum value, regardless of the state of fracture saturation.

If you wish to use Van Genuchten or Brooks–Corey functions to describe the constitutive relationships you can do so using the instructions given in Section 2.8.3.4. Unless you modify them, the default values given in Table 2.14 will be used to define the functional relationships.

Table 2.14: Default values for functions defining the discrete fracture constitutive relationships, for the Van Genuchten and Brooks–Corey models.

| Parameter | Value | Unit |
|---|---|---|
| Residual water saturation, $S_{wr}$ | 0.053 | |
| Power index (alpha), $\alpha$ | 3.5237 | $\text{m}^{-1}$ |
| Power index (beta), $\beta$ | 3.1768 | |
| Power index (gamma, computed), $\gamma$ | $1 - 1/\beta$ | |
| Pore-connectivity, $l_p$ | 0.5 | |
| Brooks–Corey exponent | $2/\beta + l_p + 2$ | |

If you wish to use tables to describe the constitutive relationships you can do so using the instructions given in Section 2.8.3.5. Unless you modify them, the default values of water saturation versus pressure head and saturation versus relative permeability listed in Table 2.15 will be used to define the tabular relationships.

Table 2.15: Default pressure-saturation and saturation-relative permeability tables for variably-saturated discrete fractures.

| Pressure (m) $\psi$ | Saturation $S_w$ |
|---|---|
| $-10.0$ | 0.053 |
| 0.0 | 1.0 |

| Saturation $S_w$ | Relative Permeability $k_{rw}$ |
|---|---|
| 0.053 | 0.053 |
| 1.0 | 1.0 |

If you wish to modify the default relationship between pressure, saturation and effective area you can do so using the instructions given below. For each instruction we will again indicate its scope (i.e. `.grok`, `.fprops`). Note that the functional relationships are always applied in a similar way to all fracture zones in the domain, while tabular relationships can vary from fracture zone to fracture zone if so desired.

The following instructions should be applied to discrete fractures, as discussed in Section 2.8.1.

## Effective area tables...End

`Scope:.grok .fprops`

Causes **grok** to use tables to describe the pressure-effective area relationship for the fracture and to begin reading a group of effective area table instructions until it encounters an `End` instruction.

By default values of contact area versus pressure head listed in Table 2.16 will be used.

Table 2.16: Default pressure-effective area table for variably-saturated discrete fractured media.

| Pressure (m) $\psi$ | Effective Area ($m^2$) |
|---|---|
| 0.0 | 1.0 |

● ● ●

These instructions are available for modifying the default pressure-effective area relationship:

## Pressure-effective area

`Scope: .grok .fprops`

1. **pressure(i), effective_area(i)...end** Pressure [L] and effective area [$L^2$].

Causes **HydroGeoSphere** to begin reading instructions which describe the pressure-contact area table that defines the relationship for the fracture. Paired values of pressure $\psi$ and effective area, which varies between 0 (full reduction) and 1 (no reduction), should be entered from lowest pressure (i.e. most negative) to highest pressure, usually zero. The last line of the table must be an End instruction, and the number of entries in the list are counted automatically to determine the table size.

• • •

## Effective area Wang-Narasimhan functions

`Scope:.grok`

Causes **HydroGeoSphere** to use the approach of Wang and Narasimhan (1985), as discussed in Section 1.1.4, for computing the pressure-effective area relationship for the fractures. These functions will automatically be applied to all fracture zones.

• • •

#### 2.8.3.3 Dual Continuum

Dual continua zones (and elements) in the domain will use a constitutive relationship based on the pseudo-soil relation developed by Huyakorn et al. (1994). Under this relationship, the medium is assigned a nodal saturation of zero above the water table and one below it. Relative permeability is applied to horizontal flow only and water travels vertically under saturated hydraulic conductivity conditions.

If you wish to use Van Genuchten or Brooks–Corey functions to describe the constitutive relationships you can do so using the instructions given in Section 2.8.3.4. Unless you modify them, the default values given in Table 2.17 will be used to define the functional relationships.

If you wish to use tables to describe the constitutive relationships you can do so using the instructions given in Section 2.8.3.5. Unless you modify them, the default values of water saturation versus pressure head and saturation versus relative permeability listed in Table 2.18 will be used to define the tabular relationships.

## Relative permeability xy

`Scope: .grok .dprops`

When using functions or tables to define the constitutive relationships, this instruction causes **HydroGeoSphere** to apply the relative permeability to horizontal flow only so that water travels vertically under saturated hydraulic conductivity conditions, similar to the behaviour of the pseudo-soil relation. This instruction should be applied to dual continua, as discussed in Section 2.8.1.

• • •

Table 2.17: Default values for functions defining the dual continua constitutive relationships, for the Van Genuchten and Brooks–Corey models.

| Parameter | Value | Unit |
|---|---|---|
| Residual water saturation, $S_{wr}$ | 0.053 | |
| Power index (alpha), $\alpha$ | 3.5237 | $\mathrm{m}^{-1}$ |
| Power index (beta), $\beta$ | 3.1768 | |
| Power index (gamma, computed), $\gamma$ | $1 - 1/\beta$ | |
| Pore-connectivity, $l_p$ | 0.5 | |
| Brooks–Corey exponent | $2/\beta + l_p + 2$ | |

Table 2.18: Default pressure-saturation and saturation-relative permeability tables for variably-saturated dual continua.

| Pressure (m) $\psi$ | Saturation $S_w$ |
|---|---|
| −10.0 | 0.053 |
| 0.0 | 1.0 |

| Saturation $S_w$ | Relative Permeability $k_{rw}$ |
|---|---|
| 0.053 | 0.053 |
| 1.0 | 1.0 |

When simulating a system with porous and dual continua, the constitutive relationships of the interface between the two continua must also be defined. The following instructions provide this functionality:

Interface unsaturated tables
Interface unsaturated van genuchten functions
Interface unsaturated brooks-corey functions
Interface relative permeability xy

Input is identical to the generic forms of the commands discussed in the following sections, except that the scope is restricted to `.dprops`.

We will now discuss instructions of a general nature which can be used to modify the default constitutive relationships for the various media, beginning with the functional relationships.

#### 2.8.3.4 Functional Constitutive Relationships

The instructions described here can be used to modify the default variably-saturated properties for a porous medium, discrete fracture or dual continuum. Before issuing them it is necessary to choose which type of medium they should be applied to, as discussed in Section 2.8.1.

For each instruction we will again indicate its scope (i.e., `.grok`, `.mprops`, `.dprops`, `.fprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g., `.mprops`) file, it will only affect the named material of which it is a part.

---

### Unsaturated brooks-corey functions...End

`Scope: .grok .mprops .fprops .dprops`

Causes **grok** to use Brooks–Corey functions (Equation 2.5) to describe the constitutive relationships for the medium and to begin reading a group of instructions that define the function until it encounters an End instruction.

• • •

---

### Unsaturated van genuchten functions...End

`Scope: .grok .mprops .fprops .dprops`

Causes **grok** to use Van Genuchten functions (Equation 2.7) to describe the constitutive relationships for the medium and to begin reading a group of instructions that define the function until it encounters an End instruction.

• • •

The previous two instructions are used to choose between the Brooks–Corey or Van Genuchten approaches for defining the functions. In either case, if no further instructions are issued, the default function parameter values given in Tables 2.12, 2.14, and 2.17 will be used for porous media, discrete fractures, and dual continuum domains, respectively. When applied to either the porous media or dual continuum domains, these instructions override the pseudo-soil default so that relative permeability factors are applied to both horizontal and vertical flow.

The following instructions can be used to modify the parameters that define the constitutive relationships:

---

### Residual saturation

`Scope: .grok .mprops .fprops .dprops`

1. **val** Residual water saturation $S_{wr}$ [-].

• • •

## Alpha

`Scope: .grok .mprops .fprops .dprops`

1. **val** Power index alpha $\alpha$ [$\mathrm{L}^{-1}$].

For the Brooks–Corey function, this parameter is computed automatically from the air entry pressure. If you use this instruction you will be prompted to enter an air-entry pressure instead and **grok** will stop.

●●●

## Beta

`Scope: .grok .mprops .fprops .dprops`

1. **val** Power index beta $\beta$ [-].

For the Van Genuchten function, this parameter must be greater than 1.0. If you enter a value less than 1.0 you will be warned and **grok** will stop. This value is used to compute $\nu$ according to Equation 2.9.

For the Brooks–Corey formulation, this value is used to recalculate the exponent in Equation 2.6 unless the instruction Exponent has been used previously for this material.

●●●

## Pore connectivity

`Scope: .grok .mprops .fprops .dprops`

1. **val** Pore connectivity $l_p$ [-].

Note that the default value of pore connectivity of 0.5 is a value recommended for the Van Genuchten formulation, so your will probably want to redefine it for the Brooks–Corey formulation. The value recommended in this case is 2.0.

For the Brooks–Corey formulation, this value is used to recalculate the exponent in Equation 2.6 unless the instruction Exponent has been used previously for this material.

●●●

## Air entry pressure

`Scope: .grok .mprops .fprops .dprops`

1. **val** Air entry pressure [L].

For the Brooks–Corey function, this value is used to compute $\alpha$ [$\mathrm{L}^{-1}$] according to Equation 2.5. For the Van Genuchten function, this parameter is not used. If you use this instruction you will be prompted to remove it and **grok** will stop.

●●●

## Exponent

`Scope: .grok .mprops .fprops .dprops`

1. **val** Exponent [-] in Equation 2.6, which is used to compute $k_r$ in the Brooks–Corey function. By default, the exponent is computed automatically from $\beta$ and $l_p$. This instruction allows you to enter a different value.

For the Van Genuchten function, this parameter is not used. If you use this instruction you will be prompted to remove it and **grok** will stop.

●●●

## Minimum relative permeability

`Scope: .grok .mprops .fprops .dprops`

1. **val** Minimum relative permeability [-].

During a simulation, the model will choose the maximum value for relative permeability between the minimum value specified here and the value computed from the active function, either Van Genuchten or Brooks–Corey. This option may improve convergence of the non-linear solution.

●●●

The following instructions can be used to generate pressure-saturation and saturation-relative permeability tables using the Van Genuchten or Brooks–Corey parameters defined for the medium. In addition to the instructions given above which define the function, these additional instructions affect the properties of the tabular data.

## Table smoothness factor

`Scope: .grok .mprops .fprops .dprops`

1. **val** Smoothness factor [-]. Smaller values cause more points to be generated for a smoother, more accurate table. The default value is $10^{-3}$.

●●●

## Table minimum pressure

`Scope: .grok .mprops .fprops .dprops`

1. **val** Minimum pressure [L] value in the pressure-saturation table. The default value is −1000 m.

• • •

## Table maximum s-k slope

`Scope: .grok .mprops .fprops .dprops`

1. **val** Maximum slope [-] of the saturation-relative permeability curve when nearing full saturation. The default value is 100.

• • •

## Generate tables from unsaturated functions

`Scope: .grok .mprops .fprops .dprops`

This command generates tables from the previously defined functions and writes the pressure-saturation data to the file *prefix*`o.p_s_table.`*material*`.dat` and saturation-relative permeability data to the file *prefix*`o.s_k_table.`*material*`.dat`, where *material* is the name of the material in the respective domain. Files are written in Tecplot ASCII format for easy visualization. The tables in the files can be copied into the respective properties file for use with the Unsaturated table command and its subcommands, described in Section 2.8.3.5.

• • •

For example, if the following Van Genuchten function parameters were defined in the `.mprops` file:

```
unsaturated van genuchten functions
    alpha
    2.25

    beta
    1.89

    residual saturation
    0.16

    minimum relative permeability
    1e-2

    table smoothness factor
    1e-2
```

```
        table minimum pressure
        -10.

        generate tables from unsaturated functions
    end ! functions
```

then the contents of the pressure-saturation output file would be:

```
    Title =  "test.mprops/porous medium"
    #  Van Genuchten function:
    #  Residual water saturation        0.160000
    #  Alpha                             2.25000
    #  Power index (beta)                1.89000
    #  Pore connectivity                0.500000
    #  Computed power index (gamma)     0.685218
    #  Minimum relative permeability               0.100000E-01
    #  Table minimum pressure                      -10.0000
    #  Table maximum s-k slope                      100.000
    #  Table smoothness factor                     0.100000E-01
    variables="Pressure","Saturation"
    zone t="Pressure - Saturation"
    #unsaturated tables
    #pressure-saturation
    -10.0000000000000      0.174869375404582
    -7.50000000000000      0.181552629607745
    -5.00000000000000      0.196301039876425
    -3.75000000000000      0.212424751627904
    -2.50000000000000      0.247430530192203
    -1.87500000000000      0.284639681867462
    -1.25000000000000      0.361026934163956
    -0.937500000000000      0.435114163471319
    -0.625000000000000      0.564528209032747
    0.000000000000000E+000   1.00000000000000
    #end ! pressure-saturation
```

The values used to define the table are included as comments, as are the instructions needed for incorporating the tabular data in the `.mprops` file. Figure 2.17 is a plot of the resulting constitutive relationships.

If desired, the `.mprops` file can be modified to use the tabular relationships. It is recommended that the Van Genuchten parameters that were used to generate the tabular data are retained in the file, either as comments or inside a Skip on...Skip off section. For our example, we could do the following:

```
    skip on
```

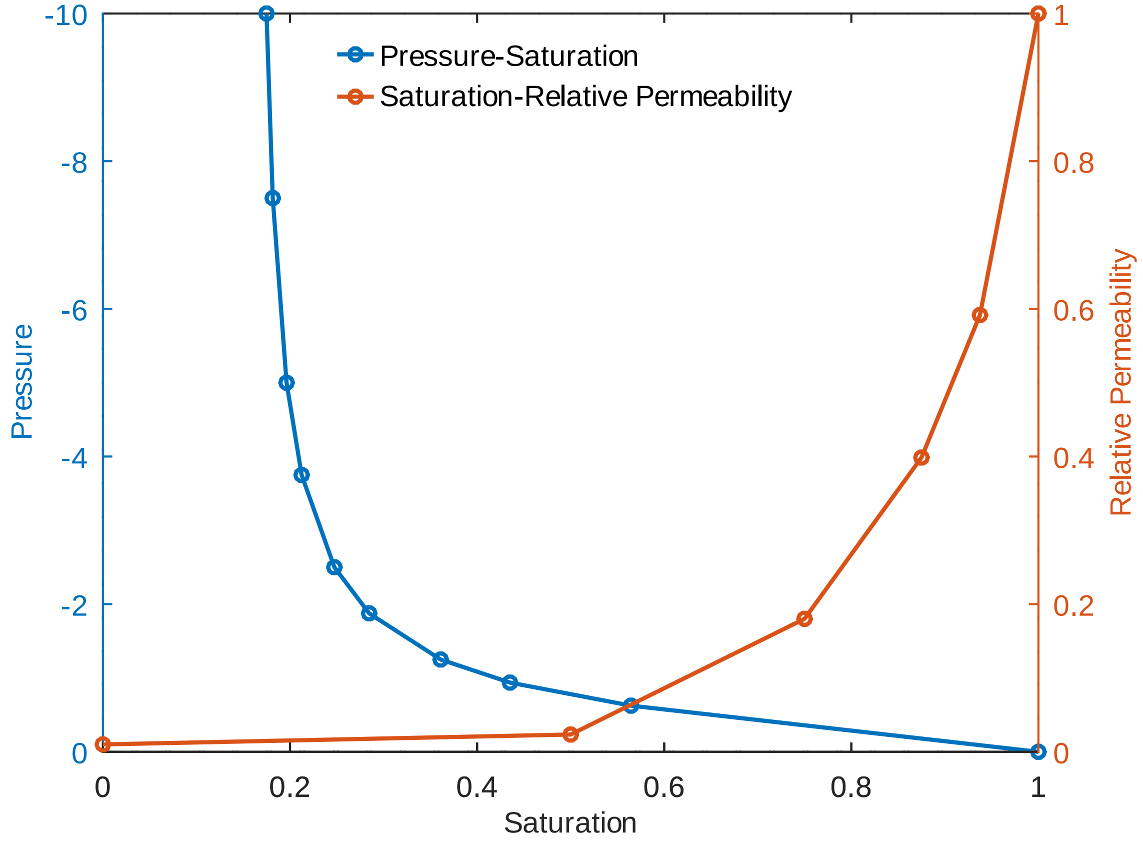


Figure 2.17: Example of using functional parameters to generate tabular constitutive relationships.

```
unsaturated van genuchten functions
...etc...
end ! functions
skip off

unsaturated tables
pressure-saturation
-10.0000000000000      0.174869375404582
-7.50000000000000      0.181552629607745
-5.00000000000000      0.196301039876425
-3.75000000000000      0.212424751627904
-2.50000000000000      0.247430530192203
-1.87500000000000      0.284639681867462
-1.25000000000000      0.361026934163956
-0.937500000000000      0.435114163471319
-0.625000000000000      0.564528209032747
0.000000000000000E+000   1.00000000000000
end ! pressure-saturation

saturation-relative k
0.000000000000000E+000  1.000000000000000E-002
0.500000000000000      2.340977494655262E-002
0.750000000000000      0.180180362276789
0.875000000000000      0.398602012758378
0.937500000000000      0.591618307040100
1.00000000000000       1.00000000000000
end ! saturation-relative k
end ! unsaturated tables
```

The following command can be used within either an Unsaturated brooks-corey functions...End or Unsaturated van genuchten functions...End command block to define tabular forms of the constitutive relationships. In this respect, it is similar to the command Generate tables from unsaturated functions, with the added benefit that it allows **grok** to be run only once and the generated tables will be used. There is no need to copy the generated tables into their respective property files followed by running **grok** again.

## Use tabulated unsaturated functions

`Scope: .grok .mprops .fprops .dprops`

This command generates tables from the previously defined functions and writes the pressure-saturation data to the file *prefix*`o.p_s_table.`*material*`.dat` and saturation-relative permeability data to the file *prefix*`o.s_k_table.`*material*`.dat`, where *material* is the name of the material in the respective domain. Files are written in Tecplot ASCII format for easy visualization. In addition, this command causes **grok** to use the generated tables to describe the constitutive relationships for the active domain. Note that when applied to either the porous media or dual continuum domains, this command overrides the pseudo-soil default so that relative permeability factors are applied to both horizontal and vertical flow.

●●●

#### 2.8.3.5 Tabular Constitutive Relationships

The instructions described here can be used to modify the default variably-saturated properties for a porous medium, discrete fracture or dual continuum. Before issuing them it is necessary to choose which type of medium they should be applied to, as discussed in Section 2.8.1.

For each instruction we will again indicate its scope (i.e., `.grok`, `.mprops`, `.dprops`, `.fprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g., `.mprops`) file, it will only affect the named material of which it is a part.

---

### Unsaturated tables

`Scope: .grok .mprops .fprops .dprops`

Causes **grok** to use tables to describe the constitutive relationships for the active domain and to begin reading a group of instructions that define the tables until it encounters an End instruction.

If no further instructions are issued, the default tabular parameter values given in Tables 2.13, 2.15, and 2.18 will be used for the porous media, discrete fractures, and dual continuum domains, respectively. When applied to either the porous media or dual continuum domains, this instruction overrides the pseudo-soil default so that relative permeability factors are applied to both horizontal and vertical flow.

●●●

The following instructions can be used within the Unsaturated tables...End command block to modify the default tables that define the constitutive relationships:

---

### Pressure-saturation

`Scope: .grok .mprops .fprops .dprops`

1. **pressure(i), saturation(i)...end** Pressure $\psi$ [L] and saturation $S$ [-].

Paired values of pressure $\psi$ and saturation $S$ should be entered from lowest pressure (i.e. most negative) to highest pressure, usually zero. The last line of the table must be an End instruction, and the number of entries in the list are counted automatically to determine the table size.

●●●

## Saturation-relative k

`Scope: .grok .mprops .fprops .dprops`

1. **saturation(i), rel_perm(i)...end** Saturation $S$ [-] and relative permeability $k_{rw}$ [-].

Paired values of saturation $S$ and relative permeability $k_{rw}$ should be entered from lowest to highest saturation. The last line of the table must be an End instruction, and the number of entries in the list are counted automatically to determine the table size.

• • •

### 2.8.4 Surface Flow

Unless you modify the default values, all zones (and elements) in the surface flow domain will be assigned the default properties listed in Table 2.19.

Table 2.19: Default properties for surface flow.

| Parameter | Value | Unit |
|---|---|---|
| Name | Surface flow defaults | |
| X friction factor | 0.0548 | $m^{-1/3}$ s |
| Y friction factor | 0.0548 | $m^{-1/3}$ s |
| Rill storage height $h_{ds}$ | $1.0 \times 10^{-6}$ | m |
| Obstruction storage height $h_o$ | $1.0 \times 10^{-6}$ | m |
| Coupling length | $1.0 \times 10^{-4}$ | m |

For each instruction we will again indicate its scope (i.e. `.grok`, `.oprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g., `.oprops`) file, it will only affect the named material of which it is a part.

## X friction

`Scope: .grok .oprops`

1. **val** Friction factor [$L^{-1/3}$ T] in the $x$-direction, $n_x$ in Equation 2.35.

• • •

## Y friction

`Scope: .grok .oprops`

1. **val** Friction factor [$L^{-1/3}$ T] in the $y$-direction, $n_y$ in Equation 2.36.

• • •

## Time varying friction

`Scope: .oprops`

1. **time(i), nmann(i)...end** Time [T] and Manning roughness coefficient [$L^{-1/3}$ T].

Reads a table of time vs Manning roughness coefficient ($n_x = n_y$) instructions until it encounters an End instruction.

• • •

## Time varying scaling of friction

`Scope: .grok .oprops`

1. **interpolate** Logical value (T/F), which if true, causes scale factors to be interpolated from the time-value table. Otherwise, the value at the left endpoint of a time panel is used.
2. **time(i), scale(i)...end** Time [T] and friction scale factor [-]. Note that times must be given in strictly increasing order.

This command uses a time series to scale the $x$ and $y$ frictions set by the commands X friction, Y friction, or Time varying friction. This command is useful for specifying conditions that override a background signal, e.g., timing of anthropogenic activities overprinted on an annual cycle of freeze-thaw.

• • •

## Time varying surface friction by temperature raster

`Scope: .grok`

1. **scale** Friction scale factor ($> 0$) [-].
2. **freezing_temp** Freezing temperature [°C].
3. **time(i), filename(i), (band(i))...end** Time [T], temperature [°C] raster filename, and optional raster band id (default value 1) list. All filenames with spaces must be enclosed by double quotes (“”) and times must be given in strictly increasing order.

In cold regions during the winter, free surface water flow may be restricted as a result of water freezing at the ground surface. Freezing reduces the conductivity of the surface domain toward water. To represent this phenomenon, the surface conductivity can be reduced by an increase in surface friction at freezing temperatures. This command uses the table of temperature raster files to increase the surface friction by the provided scaling factor when the temperatures of the chosen surface elements drop below the freezing temperature. For example:

```
time varying surface friction by temperature raster
    10000 ! scale factor
    0.0   ! freezing temperature
    ! times       raster files
    0             temperature1.asc
    2592000       temperature2.asc
    5184000       temperature3.asc
    7776000       temperature4.asc
    10368000      temperature5.asc
    12960000      temperature6.asc
end
```

Note that this command cannot be used in conjunction with the command Time varying friction.

●●●

## Rill storage height

Scope: `.grok .oprops`

1. **val** Rill storage height [L], $H_d$ in Section 2.2.2.2.

●●●

## Obstruction storage height

Scope: `.grok .oprops`

1. **val** Obstruction storage height [L], $H_o$ in Section 2.2.2.2.

●●●

## Coupling length

Scope: `.grok .oprops`

1. **val** Coupling length [L], $l_{exch}$ in Equation 2.80.

●●●

## Maximum flow depth

Scope: `.grok`

1. **val** Maximum flow depth [L], in Equations 2.46 and 2.47, the maximum of $d_o$.

●●●

## Minimum elemental energy slope

Scope: `.grok`

1. **val** Minimum elemental energy slope [-], in Equations 2.46 and 2.47, the minimum of $\partial h_o/\partial s$.

●●●

## Read rill storage from raster

Scope: `.grok`

1. **filename** Filename of raster file containing the surface elevation [L] values.
2. **scale_factor** Rill storage scaling factor [-]. Must be a strictly positive real value.

This command computes the rill storage height from the elevation values in the raster file for each element in the surface layer of the mesh. It overrides any rill storage values specified by a properties (i.e., `.oprops`) file. Computed rill storage values are written to the Tecplot ASCII file *prefix*`o.elemental_rill_storage.dat` for visualization.

●●●

The following commands set the parameters required by the rain and snowmelt boundary condition.

## Snow density

Scope: `.oprops`

1. **snow_rho** Snow density [M $L^{-3}$].

●●●

## Melting constant

Scope: `.oprops`

1. **snow_melt** Snow melting constant [M $L^{-2}$ $T^{-1}$ $^{\circ}C^{-1}$].

•••

## Sublimation constant

Scope: `.oprops`

1. **snow_sublimation** Snow sublimation constant [M $L^{-2}$ $T^{-1}$].

•••

## Threshold temperature

Scope: `.oprops`

1. **snow_threshold_temp** Snow threshold temperature [°C].

•••

## Initial snow depth

Scope: `.oprops`

1. **snow_initial_depth** Initial snow depth [L].

•••

## Initial snow depth from output file

Scope: `.grok`

1. **filename** Name of file that contains the initial snow depth [L] data.

Assigns initial snow depth values to all elements in the surface flow domain from a previously generated binary output file named *prefix*`o.snowdepth.`*suffix*, where *suffix* is a zero-padded integer identifying the output file. We recommend that this file be renamed to avoid accidentally overwriting it.

•••

### 2.8.5 Evapotranspiration

**NOTE:** Equations describing evapotranspiration in Section 2.5.3 of the Theory Manual are based on water content. However, **grok** uses saturation as input. The input required in this section is therefore saturation and not water content. The water content can be converted to saturation by dividing it by the saturated water content $\theta_s$, which is equivalent to the porosity.

Unless you modify the default values, all zones (and elements) in the ET domain will be assigned the default properties listed in Table 2.20, which are representative of a grass cover.

Table 2.20: Default properties for interception and evapotranspiration.

| Parameter | Value | Unit |
|---|---|---|
| Name | Default ET (grass) | |
| **Interception parameters:** | | |
| Canopy storage parameter $c_{int}$ | 0.0 | m |
| Initial interception storage $S_{int}^0$ | 0.0 | m |
| Canopy evaporation interval $\Delta t_{can}$ | 0.0 | s |
| **Transpiration fitting parameters:** | | |
| C1 | 0.1 | |
| C2 | 0.2 | |
| C3 | 2.0 | |
| **Transpiration limiting pressure heads:** | | |
| Wilting point $\psi_{wp}$ | $-153.0$ | m |
| Field capacity $\psi_{fc}$ | $-3.4$ | m |
| Oxic limit $\psi_o$ | 0.0 | m |
| Anoxic limit $\psi_{an}$ | 0.0 | m |
| **Evaporation limiting pressure heads:** | | |
| Minimum $\psi_{e2}$ | $-0.8$ | m |
| Maximum $\psi_{e1}$ | $-0.6$ | m |
| Leaf area index $LAI$ | 1.0 | |
| Root zone depth $L_r$ | 0.2 | m |
| Root profile $RDF$ | linear function | |
| Evaporation depth | 0.2 | m |
| Evaporation profile $EDF$ | linear function | |
| **Crop coefficients:** | | |
| Maximum crop coefficient $K_{\mathrm{c\,max}}$ | 1.0 | |
| Basal crop coefficient $K_{\mathrm{cb}}$ | 1.0 | |

For each instruction we will again indicate its scope (i.e., `.grok`, `.etprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, whereas when used in a properties (e.g., `.etprops`) file, it will only affect the named material of which it is a part.

## Time varying et zones from raster

`Scope: .grok`

1. **time(i), filename(i)...end** Time [T] and raster filename list.
2. **tecplot_output** Logical value (T/F), which if true (T) writes the time-varying ET

zones to the Tecplot ASCII file *prefix*`o.time_varying_et_zones_raster.dat`.

This command allows for time-varying ET zones by assigning zones from the time-file table. At time **time(i)** the file **filename(i)** is applied and maintained until time **time(i + 1)**. The last raster file entered in the list will be applied until the end of the simulation. For each ET domain element, the zone number is assigned from the raster cell that contains the 2-D centroid of that element. Note that ET material properties in ET properties files (`.etprops`) will not change over time, rather it is the spatial distribution of the ET zones that changes over time.

It is important to keep in mind that since zone numbers may vary over both space and time, a temporal change in zonation may require the definition of a new zone number. For example, consider a model in which there are initially five ET zones labeled $1, 2, 3, 4, 5$, where zone three corresponds to a forested area. Now suppose that at some point in the future the forest is destroyed by a fire, necessitating a change in land cover. The user would then need to create a new zone, say zone six, to define the new "burnt" land cover. In the HGS simulation, the ET zonation would then change to $1, 2, 6, 4, 5$ after the forest fire occurs. Another possibility is that only part of the forest is destroyed by the fire. In this case, there is both a spatial and temporal change in zonation. So, the part of the forest that was untouched by the fire would remain at zone three, whereas the part of the forest destroyed by the fire would be assigned zone six. The zone assignment after the fire might then look like $1, 2, 3, 6, 4, 5$. When setting up the time-file table, we suggest that a user first considers all the possible land covers that are required over the course of a simulation and assigns each one to a unique zone number. Then, each raster file in the time-file table simply contains a subset of these zone numbers.

•••

## Canopy storage parameter

`Scope: .etprops`

1. **cint_et** Canopy storage parameter [L], $c_{int}$ in Equation 2.82.

•••

In reality, rainfall and canopy evaporation are typically serial processes that occur over time periods of minutes to hours, whereas model input precipitation and potential evaporation often occur on the order of days, to weeks, to months. Therefore, it is recognized that the water balance of the canopy should mimic the underlying climate data input time interval. In cases, where the climate data input time interval is on the order of one day or longer, it is recommended to set the canopy evaporation interval equal to the climate data input time interval, e.g., daily, weekly, monthly, etc., otherwise unintended canopy storage behaviour may occur.

## Canopy evaporation interval

`Scope: .grok`

1. **time_interval** Canopy evaporation time interval [T], $\Delta t_{can}$ in Equation 2.87.

The canopy evaporation interval controls the calculation of canopy evaporation by limiting both the canopy filling rate from rainfall and the canopy evaporation rate over that time interval. Effectively, canopy storage is allowed first to fill and then empty only once per interval; see Section 2.5.3.1 of the Theory Manual for details.

●●●

## Initial interception storage

`Scope: .etprops`

1. **init_sint_et** Initial canopy interception storage value [L], $S^0_{int}$ in Equation 2.84.

●●●

The following instruction can be used to modify the default value (one for all time) of the leaf area index (LAI):

## LAI tables

`Scope: .etprops`

1. **time(i), lai(i)...end** Time [T] and leaf area index [-].

Reads a table of time and leaf area index until it encounters an End instruction. Paired values of time $t$ and leaf area index should be entered from earliest to latest time. The number of entries in the list are counted automatically to determine the table size.

Observed values of leaf area index (Scurlock et al., 2001) and maximum rooting depth (Canadell et al., 1996) for various terrestrial biomes are shown in Table 2.21.

●●●

## Time varying lai from raster

`Scope: .grok`

1. **time(i), filename(i), (band(i))...end** Time [T], leaf area index [-] raster filename, and optional raster band id (default value 1) list. All filenames with spaces must be enclosed by double quotes (“”) and times must be given in strictly increasing order.

2. **missing_data_check** Logical value (T/F), which if true (T) causes **grok** to check each raster file for missing data and issue an error if any is found.

3. **tecplot_output** Logical value (T/F), which if true (T) writes time-varying LAI values to the Tecplot ASCII file *prefix*`o.time_varying_lai_raster.dat`.

This command allows for time-varying leaf area index values from a time-file table. For each ET domain element, the leaf area index is interpolated at the 2-D centroid of that element from the raster files at the endpoints of the time window that contains the current simulation time. Note that this command overrides any leaf area index values set by any ET properties file (`.etprops`).

●●●

## Transpiration fitting parameters

`Scope: .etprops`

1. **C_1** Coefficient $C_1$ [-] in Equation 2.91.
2. **C_2** Coefficient $C_2$ [-] in Equation 2.91.
3. **C_3** Coefficient $C_3$ [-] in Equation 2.93. By default, this coefficient is set to one, which gives a linear ramping function whereas higher values would give higher order ramping functions.

●●●

## Transpiration limiting saturations

`Scope: .etprops`

1. **thwp_et** Saturation [-] at wilting point, equal to $\theta_{wp}/\theta_s$, with $\theta_{wp}$ used in Equation 2.93.
2. **thfc_et** Saturation [-] at field capacity, equal to $\theta_{fc}/\theta_s$, with $\theta_{fc}$ used in Equation 2.93.
3. **tho_et** Saturation [-] at oxic limit, equal to $\theta_o/\theta_s$, with $\theta_o$ used in Equation 2.93.
4. **than_et** Saturation [-] at anoxic limit, equal to $\theta_{an}/\theta_s$, with $\theta_{anp}$ used in Equation 2.93.

●●●

## Transpiration limiting pressure head

`Scope: .etprops`

1. **Hwp_et** Pressure head [L] at wilting point, $\psi_{wp} = \psi(\theta_{wp})$.
2. **Hfc_et** Pressure head [L] at field capacity, $\psi_{fc} = \psi(\theta_{fc})$.
3. **Ho_et** Pressure head [L] at oxic limit, $\psi_o = \psi(\theta_o)$.

4. **Han_et** Pressure head [L] at anoxic limit, $\psi_{an} = \psi(\theta_{an})$.

This is an alternative to the command Transpiration limiting saturations.

●●●

## Evaporation limiting saturations

`Scope: .etprops`

1. **the2_et** Saturation [-] below which evaporation is zero, equal to $\theta_{e2}/\theta_s$, with $\theta_{e2}$ used in Equation 2.101.
2. **the1_et** Saturation [-] above which full evaporation can occur, equal to $\theta_{e1}/\theta_s$, with $\theta_{e1}$ used in Equation 2.101.

●●●

## Evaporation limiting pressure head

`Scope: .etprops`

1. **He2_et** Pressure head [L] below which evaporation is zero, $\psi_{e2} = \psi(\theta_{e2})$.
2. **He1_et** Pressure head [L] above which full evaporation can occur, $\psi_{e1} = \psi(\theta_{e1})$.

This is an alternative to the command Evaporation limiting saturations.

●●●

The following command can be used to limit the amount of surface evaporation.

## Limit surface evaporation to percentage of PET

`Scope: .etprops`

1. **pet_percent** Percentage of PET (range $[0, 1]$).

Limits the amount of surface evaporation to the specified percentage of PET prescribed by the potential evapotranspriation boundary condition. By default, no limit is applied.

●●●

## Root depth

`Scope: .etprops`

1. **max_root_depth** Maximum root depth ($L_r$) [L].

Table 2.21: Observed values of leaf area index (Scurlock et al., 2001) and maximum rooting depth (Canadell et al., 1996) for various terrestrial biomes.

| Vegetative cover | Leaf area index | Maximum rooting depth (m) |
|---|---|---|
| Desert | 1.31 | 9.5 |
| Tundra, circumpolar and alpine | 2.69 | 0.5 |
| Wetlands, temperate and tropical | 6.34 | |
| Grasslands, temperate | 2.50 | 2.6 |
| Grasslands, tropical | 2.50 | 15.0 |
| Crops, temperate and tropical | 4.22 | 2.1 |
| Shrubland, heath or Mediterranean-type vegetation | | 5.2 |
| Forest: | | |
| boreal deciduous broadleaf | 2.64 | 2.0 |
| boreal evergreen needleleaf | 3.50 | 2.0 |
| boreal/temperate deciduous needleleaf | 4.63 | 2.0 |
| temperate deciduous broadleaf | 5.12 | 2.9 |
| temperate evergreen needleleaf | 6.70 | 3.9 |
| temperate evergreen broadleaf | 5.82 | |
| tropical deciduous broadleaf | 3.92 | 3.7 |
| tropical evergreen broadleaf | 4.90 | 7.3 |
| Plantations (managed forests) | 8.72 | |
| Overall mean | 5.23 | |

Specifies a fixed maximum root depth.

●●●

Root length density can be defined via polynomial functions of relative depth ($z_r = z/L_r$) that are mapped onto porous media elements above the maximum root depth ($L_r$). These functions are defined so that the area under their graph is one. Four options are available: constant, linear, quadratic, and cubic; see Figure 2.18. Root length density may also be defined by a piecewise linear function of relative depth that is represented by a table. By default, root length density is defined by the linear polynomial function. The following commands may be used to select one of the other functions:

Rdf constant function
Rdf quadratic decay function
Rdf cubic decay function

The following command may be used to specify the root length density via a table.

## Rdf table

`Scope: .etprops`

1. **depth(i), density(i)...end** Relative depth [-] and root length density [L L$^{-3}$] table.

Defines root length density as a function of relative depth via a table. Depth values must be belong to the interval $[0, 1]$. Density values are automatically normalized so that the area under the table is equal to one. Any depth values that fall below the minimum or above the maximum depth values in the table are assigned a density value of zero.

●●●

Root growth can be defined by a table using the command:

## Time-root depth table

`Scope: .etprops`

1. **time(i), root_depth(i)...end** Time [T] and root depth [L] table.

Reads a root depth time series until it encounters an End instruction.

●●●

Root growth can also be defined by a logistic growth function via the following commands:

## Root growth...End

`Scope: .etprops`

**grok** reads instructions that define a root growth function until it encounters an End

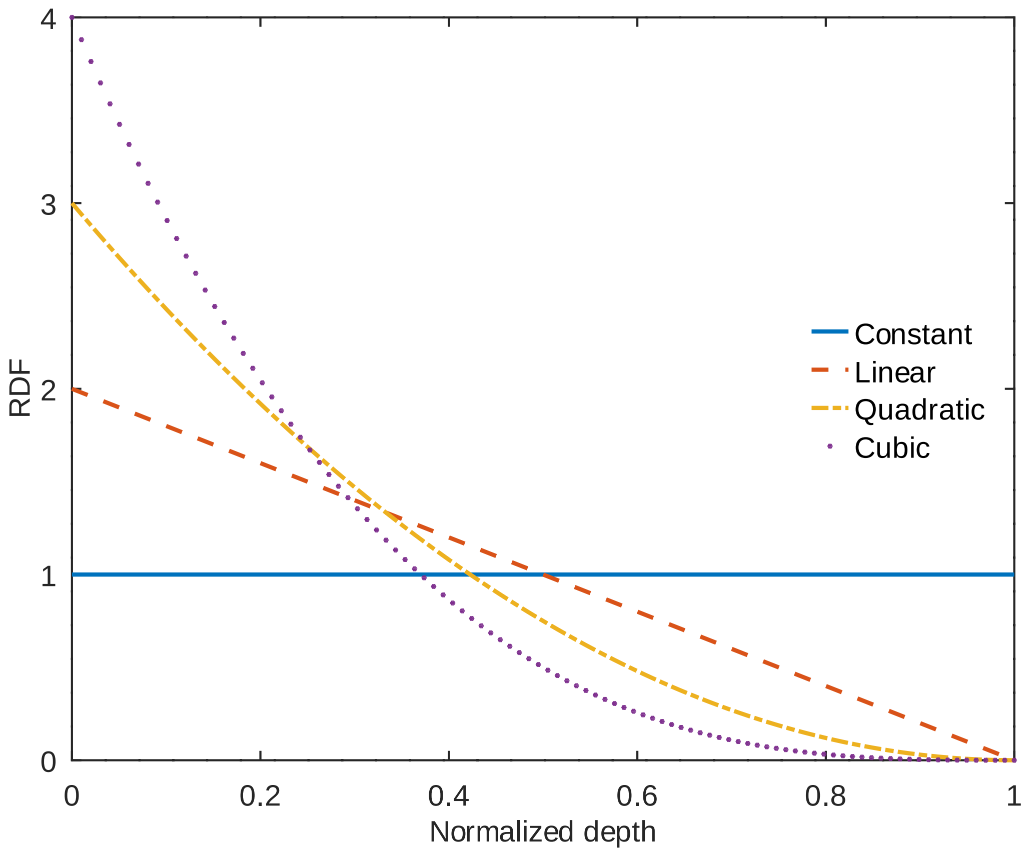


Figure 2.18: Normalized root depth functions.

instruction.

•••

---

## Root growth period

`Scope: .etprops`

1. **Rg_period** Period for vegetation growth [T], typically one year.

•••

---

## Growth beginning time

`Scope: .etprops`

1. **Rg_begin** Start time of root growth [T].

•••

---

## Harvest time

`Scope: .etprops`

1. **Rg_harvest** Crop harvest time [T].

•••

---

## Initial root depth

`Scope: .etprops`

1. **Rg_initdepth** Initial crop root depth [L].

•••

---

## Maximum root depth

`Scope: .etprops`

1. **Rg_maxdepth** Maximum crop root depth [L].

•••

---

## Verhulst-Pearl growth, time-depth

`Scope: .etprops`

1. **Rg_vptime, Rg_vpdepth** Time [T] and crop root depth [L] data that is used to derive a Verhulst–Pearl growth function.

●●●

## Verhulst-Pearl growth, 50% max depth

`Scope: .etprops`

For Verhulst–Pearl growth, 50% of the maximum crop root depth is assumed to be reached after 50% of the growing season.

●●●

The following is an example of defining root growth via a logistic growth function.

```
root growth
! option #1
!    time-root depth table
!    end
! option #2
   root growth period
365.0
growth beginning time
160.0
harvest time
250.0
initial root depth
0.1
maximum root depth
80.0
!   verhulst-pearl growth, time-depth
!   205.0  40.0
! option #3
 verhulst-pearl growth, 50\% max depth
end
```

## Evaporation depth

`Scope: .etprops`

1. **evap_depth** Evaporation depth [L].

Evaporation as a function of depth is treated in a similar fashion as the root zone depth described above.

●●●

By default, the linear form of the evaporation function is used. The following instructions are available for using the other forms:

Edf constant function
Edf quadratic decay function
Edf cubic decay function

## Potential evaporation using transpiration

`Scope: .grok`

With this instruction potential evaporation is calculated using Equation 2.99 in the Theory Manual. By default potential evaporation is calculated from Equation 2.103 in the Theory Manual.

• • •

The following command PET from crop coefficients based on the FAO-56 guideline document (Allen et al., 1998), will override the default behavior set by the command Transpiration fitting parameters (Kristensen and Jensen, 1975) Equation (2.90) in the Theory manual. The crop coefficient method is intended to use reference evapotranspiration for a well-watered grass surface (Allen et al., 1998) as model input for potential evapotranspiration, $E_p$. When the crop coefficient formulation is used, LAI is not included in the evaporation and transpiration partitioning, but LAI is still used in the calculation of canopy evaporation, $E_{can}$. Instead, a time-value table for the basal crop coefficient ($K_{\mathrm{cb}}$ in Allen et al. (1998)) is used to capture the time-varying nature of seasonal crop growth, while a time-value table is used to specify the maximum crop coefficient ($K_{\mathrm{c\,max}}$ in Allen et al. (1998)) for determination of potential evaporation of intermittently moist soils resulting from rainfall or irrigation.

## PET from crop coefficients

`Scope: .etprops`

This command partitions potential evapotranspiration, $E_p$, in potential evaporation (PE) and potential transpiration (PT) such that

$$\mathrm{PT} = (E_p - E_{can})K_{\mathrm{cb}}$$

$$\mathrm{PE} = (E_p - E_{can})(K_{\mathrm{c\,max}} - K_{\mathrm{cb}})$$

where $E_{can}$ is the canopy evaporation, $K_{\mathrm{c\,max}}$ is the maximum crop coefficient, and $K_{\mathrm{cb}}$ is the basal crop coefficient. Users have the option of specifying time-varying crop coefficients, otherwise, they are given by their default values in Table 2.20.

• • •

## Maximum crop coefficient table

`Scope: .etprops`

1. **time(i), Kcmax(i)...end** Time [T] and maximum crop coefficient [-].

Specifies a time-value table for the maximum crop coefficient $K_{\mathrm{c\,max}}$.

●●●

## Basal crop coefficient table

Scope: `.etprops`

1. **time(i), Kcb(i)...end** Time [T] and basal crop coefficient [-].

Specifies a time-value table for the basal crop coefficient $K_{\mathrm{cb}}$.

●●●

## Echo et at point

Scope: `.grok`

1. **x1, y1** *xy*-coordinates [L] of the point.

This instruction finds the column of nodes that the given coordinate falls within and then writes pertinent evapotranspiration information to the ET PROPERTIES section of *prefix*-`o.eco` file. This command should be issued only after all ET zones have been assigned property values. For example:

```
ET properties for element column
x = 460000.0000000000, y = 6350000.000000000
Element: 725644

ZONE:  1
 ET MATERIAL: et1
      Canopy storage parameter                =   0.000000000000000E+000
      Transpiration fitting parameters:
          C1                                  =   0.300000000000000
          C2                                  =   0.200000000000000
          C3                                  =   3.000000000000000E-006
      Transpiration limiting saturations:
          Wilting point                       =   0.200000000000000
          Field capacity                      =   0.320000000000000
          Oxic limit                          =   0.760000000000000
          Anoxic limit                        =   0.900000000000000
      Evaporation limiting saturations:
          Minimum (no evaporation below this)  =   0.200000000000000
          Maximum (full evaporation above this) =   0.320000000000000
      Initial interception storage            =   0.000000000000000E+000
  TABULAR DATA:
```

```
Leaf Area Index (LAI):
 Time               LAI
  0.00000       2.08000
 0.100000E+42   2.08000

Maximum evaporative zone depth             =    3.00000000000000
Evaporation Function EF:
 Normalized Depth               EF
  0.00000                  3.00000
 0.500000E-01              2.70750
 0.100000                  2.43000
 0.150000                  2.16750
 0.200000                  1.92000
 0.250000                  1.68750
 0.300000                  1.47000
 0.350000                  1.26750
 0.400000                  1.08000
 0.450000                 0.907500
 0.500000                 0.750000
 0.550000                 0.607500
 0.600000                 0.480000
 0.650000                 0.367500
 0.700000                 0.270000
 0.750000                 0.187500
 0.800000                 0.120000
 0.850000                 0.675000E-01
 0.900000                 0.300000E-01
 0.950000                 0.750000E-02
  1.00000                  0.00000

Maximum root zone depth                    =    3.00000000000000
Evaporation Function RF:
 Normalized Depth               RF
  0.00000                  3.00000
 0.500000E-01              2.70750
 0.100000                  2.43000
 0.150000                  2.16750
 0.200000                  1.92000
 0.250000                  1.68750
 0.300000                  1.47000
 0.350000                  1.26750
 0.400000                  1.08000
 0.450000                 0.907500
 0.500000                 0.750000
 0.550000                 0.607500
```

```
     0.600000                 0.480000
     0.650000                 0.367500
     0.700000                 0.270000
     0.750000                 0.187500
     0.800000                 0.120000
     0.850000                 0.675000E-01
     0.900000                 0.300000E-01
     0.950000                 0.750000E-02
      1.00000                  0.00000

 ELEMENT    ET FLAG    DEPTH         EDF           RDF
 3529          T       0.00000       0.703704      0.703704
 3189          F       1.00000       0.259259      0.259259
 2849          F       2.00000       0.370370E-01  0.370370E-01
 2509          F       3.00000       0.00000       0.00000
 2169          F       4.00000       0.00000       0.00000
 1829          F       5.00000       0.00000       0.00000
 1489          F       6.00000       0.00000       0.00000
 1149          F       7.00000       0.00000       0.00000
 809           F       8.00000       0.00000       0.00000
 469           F       9.00000       0.00000       0.00000
 129           F       10.0000       0.00000       0.00000
                                   ------------  ------------
Totals                               1.00000       1.00000
```

In this case, the element column falls in ET zone 1, and quadratic decay functions for root depth and evaporation depth have been specified. Because the total depth of the column (10 m) exceeds the maximum evaporative zone depth (3 m) and the maximum root zone depth (3 m), all of the transpiration and evaporative potentials have been distributed to the element column, as indicated by total RDF and EDF values of 1.0. The RDF and EDF values would be less than 1.0 if the total depth of the column was less than 3 m.

•••

### 2.8.6 Transport

#### 2.8.6.1 Porous Medium

By default, all porous media zones (and elements) in the domain will be assigned default porous media transport properties which are listed in Table 2.22. Included here are parameters for modifying the porous medium so that it acts as a double-porosity medium for simulating transport, as described in Section 2.7.1.3 of the Theory Manual and also for isotopic fractionation, as described in Section 2.7.1.4 of the Theory Manual.

The following instructions can be applied to porous media, as discussed in Section 2.8.1, to

Table 2.22: Default values for porous media transport properties.

| Parameter | Value | Unit |
|---|---|---|
| Longitudinal dispersivity $\alpha_l$ | 1.0 | m |
| Horizontal component of transverse dispersivity $\alpha_t$ | 0.1 | m |
| Vertical component of transverse dispersivity $\alpha_t$ | 0.1 | m |
| Bulk density $\rho_b$ | 2650.0 | kg m$^{-3}$ |
| Tortuosity $\tau$ | 0.1 | |
| Immobile zone porosity $\theta_{\mathrm{Imm}}$ | 0.0 | |
| Immobile zone mass transfer coefficient $\alpha_{\mathrm{Imm}}$ | 0.0 | s$^{-1}$ |
| Reverse fractionation rate $k_r$ | 0.0 | s$^{-1}$ |
| Fractionation factor $\alpha_r$ | 0.0 | |
| Mass ratio, solid to water phases $x_r$ | 0.0 | |
| Thermal conductivity of the solids $k_s$ | 3.0 | W m$^{-1}$ K$^{-1}$ |
| Specific heat capacity of the solids $c_s$ | 738.0 | J kg$^{-1}$ K$^{-1}$ |

modify the default transport parameters. For each instruction we will indicate its scope (i.e. `.grok .mprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.mprops`) file, it will only affect the named material of which it is a part.

## Longitudinal dispersivity

`Scope: .grok .mprops`

1. **val** Longitudinal dispersivity [L], $\alpha_l$ in Equation 2.122.

•••

## Transverse dispersivity

`Scope: .grok .mprops`

1. **val** Horizontal component of the transverse dispersivity [L], $\alpha_t$ in Equation 2.122.

•••

## Vertical transverse dispersivity

`Scope: .grok .mprops`

1. **val** Vertical component of the transverse dispersivity [L], $\alpha_t$ in Equation 2.122.

•••

## Tortuosity

Scope: `.grok .mprops`

1. **val** Tortuosity [-], $\tau$ in Equation 2.122.

•••

## Read elemental tortuosity from file

Scope: `.grok`

1. **filename** Name of the file which contains the variable tortuosity data.

The input file should contain the following data:

1. **element_number, tort** Element number, tortuosity [-].

Data is read from the file until end-of-file is reached. The user can supply variable values of tortuosity for any number of elements in the input file. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the binary output file *prefix*`o.ElemTort_pm.0001`.

•••

## Read elemental tortuosity from binary file

Scope: `.grok`

1. **filename** Name of the binary file that contains the tortuosity values.

The file must be formatted as follows:

1. **bom** Byte order mark, the number 32,767 (2-byte signed integer).
2. **nelems** Number of elements being assigned values (4-byte signed integer).
3. **elemIds(i), i=1,nelems** Element numbers (4-byte signed integer).
4. **tort(i), i=1,nelems** Tortuosity [-] values (8-byte real).

Assigns tortuosity values from the input file to the specified set of elements. Any previously set zoned values or user specified element-variable values will be honoured. Note that if the input file is being generated by Fortran, then it must be opened with the access specifier `access='stream'` to avoid the 4-byte record headers that are added when writing with sequential access. Conversion to the correct endianness is handled automatically when reading the file.

• • •

## Anisotropic tortuosity ratio

`Scope: .grok .mprops`

1. **y_tortratio** Tortuosity ratio [-] in the $y$-direction. Default value is 1.
2. **z_tortratio** Tortuosity ratio [-] in the $z$-direction. Default value is 1.

By default, tortuosity is isotropic, since the ratio values are set to one in both the $y$- and $z$-directions. You can make tortuosity anisotropic by entering a value in the range $(0, 1)$. These values will be used to multiply the tortuosity, $\tau$, in the $y$- and $z$-directions, respectively, to obtain the directional values.

• • •

## Bulk density

`Scope: .grok .mprops`

1. **val** Bulk density of dry soil [M L$^{-3}$], $\rho_b$ in Equation (2.121).

• • •

Note that the density of the solid phase of the porous medium, $\rho_s$, is computed automatically from the bulk density and porosity via the following equation

$$\rho_s = \frac{\rho_b}{1 - \theta_s} \, .$$

By default, the porous medium acts as a single-porosity medium (i.e., the immobile zone is inactive) because the porosity and mass transfer coefficient are set to zero. In order to activate the double porosity feature, you can enter non-zero values for these parameters using the following two instructions:

## Immobile zone porosity

`Scope: .grok .mprops`

1. **val** Immobile zone porosity [-], $\theta_{\text{Imm}}$ in Equations (2.126, 2.139, 2.140).

• • •

## Immobile zone mass transfer coefficient

`Scope: .grok .mprops`

1. **val** Immobile zone mass transfer coefficient [T$^{-1}$], $\alpha_{\mathrm{Imm}}$ in Equation 2.138.

• • •

## Isotope fractionation data...End

Scope: `.mprops`

Causes **grok** to begin reading a group of isotope fractionation instructions until it encounters an End instruction.

If no further instructions are issued, the default isotopic fractionation parameter values listed in Table 2.22 will be used.

• • •

The following three instructions can be used to change these values:

## Reverse rate

Scope: `.mprops`

1. **val** Reverse fractionation rate [L$^{-1}$], $k_r$ in Equation 2.141.

• • •

## Fractionation factor

Scope: `.mprops`

1. **val** Fractionation factor [-], $\alpha_r$ in Equation 2.141.

• • •

## Rock-water mass ratio

Scope: `.mprops`

1. **val** Isotopic rock-water mass ratio [-], $x_r$ in Equation 2.127.

• • •

The next instructions can be used to change the thermal properties of the porous medium:

## Thermal conductivity of solid

Scope: `.grok .mprops`

1. **val** Temperature invariant thermal conductivity of the solids [M L T$^{-3}$ K$^{-1}$]. The bulk thermal conductivity is computed internally from the volume fractions of the solid and liquid phases.

• • •

## Temperature-dependent thermal conductivity of solid

`Scope: .grok .mprops`

1. **k_s1** Thermal conductivity [M L $T^{-3}$ $K^{-1}$] at temperature **t_s1**.
2. **t_s1** Temperature [°C] at which the thermal conductivity is equal to **k_s1**.

If this instruction is specified, then the thermal conductivity of the solid phase is temperature dependent. The bulk thermal conductivity is also temperature dependent and is computed internally from the volume fractions of the solid and liquid phases. It is assumed here that the thermal conductivity of the solids decreases at a constant rate of 1% per 10 K increase in temperature and the relationship between thermal conductivity and temperature is defined by the pair of values (**k_s1**, **t_s1**).

• • •

## Geometric bulk thermal conductivity relation

`Scope:  .mprops`

The default equation used to compute the bulk thermal conductivity is Equation 2.157. If that instruction is specified, the thermal conductivity of the porous medium is instead computed with Equation 2.160.

• • •

## Nonlinear bulk thermal conductivity relation

`Scope:  .mprops`

1. **k_sat** Thermal conductivity of the saturated porous medium [M L $T^{-3}$ $K^{-1}$].
2. **k_dry** Thermal conductivity of the dry porous medium [M L $T^{-3}$ $K^{-1}$].
3. **k_C** Fitting parameter [-].

The default equation used to compute the bulk thermal conductivity is Equation 2.157. If that instruction is specified, the thermal conductivity of the porous medium is instead computed with Equation 2.161.

• • •

## Specific heat capacity of solid

`Scope: .grok .mprops`

1. **val** Specific heat capacity of the solid phase [$L^2$ $T^{-2}$ $K^{-1}$].

●●●

#### 2.8.6.2 Discrete Fractures

By default, all fracture zones (and elements) in the domain will be assigned default transport properties which are listed in Table 2.23.

Table 2.23: Default values for discrete fracture transport properties.

| Parameter | Value | Unit |
|---|---|---|
| Longitudinal dispersivity $\alpha_l$ | 1.0 | m |
| Transverse dispersivity $\alpha_t$ | 1.0 | m |
| Coupling dispersivity | 0.0 | m |

The following instructions can be applied to discrete fractures, as discussed in Section 2.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e., `.grok .fprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g., `.fprops`) file, it will only affect the named material of which it is a part.

## Longitudinal dispersivity

`Scope: .grok .fprops`

1. **val** Longitudinal dispersivity [L], similar to $\alpha_l$ in Equation 2.122.

●●●

## Transverse dispersivity

`Scope: .grok .fprops`

1. **val** Transverse dispersivity [L], similar to $\alpha_t$ in Equation 2.122.

●●●

## Coupling dispersivity

`Scope: .grok .fprops`

1. **val** Dispersivity value [L] to be applied during exchange between the discrete fracture and subsurface domains.

•••

#### 2.8.6.3 Dual Continuum

By default, all dual continua zones (and elements) in the domain will be assigned default transport properties which are listed in Table 2.24.

The following instructions can be applied to dual continua, as discussed in Section 2.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e. `.grok .dprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.dprops`) file, it will only affect the named material of which it is a part.

---

## Longitudinal dispersivity

`Scope: .grok .dprops`

1. **val** Longitudinal dispersivity [L], $\alpha_{ld}$ in Equation 2.130.

•••

---

## Transverse dispersivity

`Scope: .grok .dprops`

1. **val** Horizontal component of the transverse dispersivity [L], $\alpha_{td}$ in Equation 2.130.

•••

---

## Vertical transverse dispersivity

`Scope: .grok .dprops`

1. **val** Vertical component of the transverse dispersivity [L], $\alpha_{td}$ in Equation 2.130.

Table 2.24: Default values for dual-continua transport properties.

| Parameter | Value | Unit |
|---|---|---|
| Longitudinal dispersivity $\alpha_{ld}$ | 1.0 | m |
| Horizontal component of transverse dispersivity $\alpha_{td}$ | 0.1 | m |
| Vertical component of transverse dispersivity $\alpha_{td}$ | 0.1 | m |
| Bulk density $\rho_{bd}$ | 2650.0 | kg m$^{-3}$ |
| Tortuosity $\tau_d$ | 0.1 | |
| First-order mass transfer coefficient $\alpha_s$ | 0.0 | s$^{-1}$ |

• • •

### Tortuosity

Scope: `.grok .dprops`

1. **val** Tortuosity [-], $\tau_d$ in Equation 2.130.

• • •

### Anisotropic tortuosity ratio

Scope: `.grok .dprops`

1. **y_tortratio** Tortuosity ratio [-] in the $y$-direction. Default value is 1.
2. **z_tortratio** Tortuosity ratio [-] in the $z$-direction. Default value is 1.

By default, tortuosity is isotropic, since the ratio values are set to one in both the $y$- and $z$-directions. You can make tortuosity anisotropic by entering a value in the range $(0, 1)$. These values will be used to multiply the tortuosity, $\tau_d$, in the $y$- and $z$-directions, respectively, to obtain the directional values.

• • •

### Bulk density

Scope: `.grok .dprops`

1. **val** Bulk density [M L$^{-3}$], $\rho_{bd}$ in Equation 2.129.

• • •

### First-order mass exchange

Scope: `.dprops`

1. **val** First-order mass transfer coefficient [T$^{-1}$], $\alpha_s$ in Equations 2.143 and 2.144.

• • •

#### 2.8.6.4 Surface Runoff

By default, all surface flow zones (and elements) in the domain will be assigned default transport properties which are listed in Table 2.25.

Table 2.25: Default values for surface flow transport properties.

| Parameter | Value | Unit |
|---|---|---|
| Longitudinal dispersivity $\alpha_{lo}$ | 1.0 | m |
| Transverse dispersivity $\alpha_{to}$ | 1.0 | m |
| Coupling dispersivity | 0.0 | m |

The following instructions can be applied to the surface flow domain, as discussed in Section 2.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e. `.grok`, `.oprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.oprops`) file, it will only affect the named material of which it is a part.

## Longitudinal dispersivity

`Scope: .grok .oprops`

1. **val** Longitudinal dispersivity [L], $\alpha_{lo}$. Analogous to $\alpha_l$ in Equation 2.122.

•••

## Transverse dispersivity

`Scope: .grok .oprops`

1. **val** Horizontal component of the transverse dispersivity [L], $\alpha_t$. Analogous to $\alpha_t$ in Equation 2.122.

•••

## Coupling dispersivity

`Scope: .grok .oprops`

1. **val** Dispersivity value [L] to be applied during exchange between the overland flow and subsurface domains. Default value is 0.0.

•••

## Dry albedo

`Scope: .grok .oprops`

1. **Value** Dry albedo [-] for soil type, $\alpha_{dry}$ in Equation 2.171. Default value is 0.35.

•••

## Saturated albedo

Scope: `.grok .oprops`

1. **Value** Saturated albedo [-] for soil type, $\alpha_{sat}$ in Equation 2.171. Default value is 0.18.

•••

## Heat coupling length

Scope: `.grok .oprops`

1. **Value** Heat coupling length [L]. The depth of the surface/subsurface exchange zone used to compute $\alpha_o$ in Equation 2.182. Default value is $10^{-4}$ m.

•••

## 2.9 Winter Processes

### 2.9.1 Groundwater Flow with Freezing and Thawing of Pore Water

The following commands can be used to control the freezing and thawing of pore water in the porous medium and dual continuum domains (see Sections 2.6.2–2.6.3).

## Pore water freezing-thawing...End

Defines the pore water freezing and thawing parameters for the porous medium and dual continuum domains until it encounters an End instruction. Note that all temperatures must be specified in terms of degrees Celsius. For example:

```
pore water freezing-thawing
    surface temperature
        0              -7.0
        2592000        -6.1
        5184000        -0.9
        7776000         6.2
        10368000       12.9
        12960000       18.0
        15552000       20.2
        18144000       19.3
        20736000       14.8
        23328000        8.6
        25920000        2.4
        28512000       -4.0
        31104000       -7.0
    end

    thermal diffusivity
    2.0e-7

    background temperature
    6.0

    melting temperature
    -0.5

    freezing temperature
    0.5

    maximum freezing depth
    2.0

    integration convergence criteria
    0.01

    memory length for convolution integral
    15552000.0

    interpolate surface temperature
end
```

• • •

The Pore water freezing-thawing...end command is composed of the following subcommands:

## Surface temperature

1. **time(i), val(i)...end** Time [T] and surface temperature [°C] value list.

At time **time(i)** the value **val(i)** is applied and maintained until time **time(i + 1)**. The last value entered in the list will be applied until the end of the simulation. Note that this command cannot be used in conjunction with the command Surface temperature from raster.

• • •

## Surface temperature from raster

1. **time(i), filename(i)...end** Time [T] and raster filename list.

At time **time(i)** the raster file **filename(i)** is applied and maintained until time **time(i+1)**. The last raster file entered in the list will be applied until the end of the simulation. **HydroGeoSphere** uses the nodal *xy*-coordinate to interpolate a value for the surface temperature [°C] at each time and location. Note that this command cannot be used in conjunction with the command Surface temperature.

• • •

## Thermal diffusivity

1. **thermal_diff** Thermal diffusivity [$L^2$ $T^{-1}$]. Default value of $2 \times 10^{-7}$ $m^2$ $s^{-1}$.

• • •

## Background temperature

1. **background_temp** Background temperature [°C].

• • •

## Melting temperature

1. **melting_temp** Melting temperature [°C].

• • •

## Freezing temperature

1. **freezing_temp** Freezing temperature [°C].

• • •

## Maximum freezing depth

1. **freezing_depth** Maximum freezing depth [L].

•••

## Integration convergence criteria

1. **tol** Integration convergence criteria [$\mathrm{T}^{-1/2}$ °C]. Default value of 0.01 $\mathrm{s}^{-1/2}$ °C.

Controls the approximation quality of the integral that defines the porous medium temperature. If $I_n$ is the approximation of the integral on $n$ points and $I_{2n}$ is the approximation of the integral on $2n$ points, then convergence of this approximation is defined by

$$|I_{2n} - I_n| \leq \mathrm{tol},$$

where $n = 2^k n_0$ for $k = 0, 1, 2, \ldots$.

•••

## Integration maximum iterations

1. **maxits** Maximum number of iterations for integration convergence. Default value of 20.

Terminates the approximation of the integral that defines the porous medium temperature after maxits iterations and issues a warning message to the screen and the *prefix*`o.lst` file.

•••

## Memory length for convolution integral

1. **memory_length** Memory length for convolution integral [T]. Default value of 15 552 000 s (6 months).

The memory length of the convolution integral can be set to control the amount of thermal memory in the system. After a certain amount of time relative to the past, there should be no influence on the present value of the porous medium temperature. In practice, the memory length of the convolution integral should be set to half the surface temperature cycle length. For example, if seasonally varying ground surface temperature is cyclic over a period of a year, then an appropriate memory length is 6 months (in consistent time units). If a diurnal temperature cycle was being simulated, then an appropriate memory length would be 12 hours.

•••

### Interpolate surface temperature

Causes the surface temperature to be interpolated over the time window that contains the current simulation time. Otherwise, by default, the surface temperature is held constant over the time window and is equal to the temperature at the left endpoint of the window.

●●●

The following commands can be used to control freezing and thawing in the dual continuum domain.

### No freezing in dual domain

Turns off freezing in the dual domain. Note that this command cannot be used in conjunction with the command Dual freezing by pm temperature.

●●●

### Dual freezing by pm temperature

1. **transfer_coeff** Thermal transfer coefficient [T °C$^{-1}$].

Makes the freezing rate of the dual domain a linear function of porous medium temperature below the freezing temperature. The slope of the linear relationship is given by the thermal transfer coefficient. Note that this command cannot be used in conjunction with the command No freezing in dual domain.

●●●

## 2.10 Hydraulic Mixing Cell

***Hydraulic Mixing Cells are not available with all versions of HydroGeoSphere. Contact sales@aquanty.com for more information on this functionality.***

The Hydraulic Mixing Cell (HMC) flow tracking tool allows **HydroGeoSphere** to track the flow of water from distinct, user-defined sources throughout the model domain (Partington, 2013; Partington et al., 2011, 2012, 2013; Schilling et al., 2017). After defining or "tagging" hydrologic sources to the model, the HMC algorithm uses the flow solution at each timestep to track the fraction of water from each designated source as it moves through the model domain.

When HMC flow tracking is activated, all compatible flow boundary conditions in the model are automatically tagged for tracking by default. In addition, user-defined zones can be specified in each model domain (porous medium, surface, and channel) based on node selections. The water assoicated with the initial conditions for these user-defined zones will then be tagged for tracking. The initial water associated with these user-defined zones and water that enters through compatible flow boundary conditions are considered distinct water sources.

Hydraulic Mixing Cell flow tracking output is recorded at user-defined locations similar to the command Set hydrograph nodes. Output can be generated at multiple locations within various model domains and is based on node selections.

Hydraulic Mixing Cell flow tracking is available for transient flow simulations and can also be used in conjunction with the command Defined flow. Output is generated for each specified output location and is written to the Tecplot ASCII file named *prefix*`o.hmc_fractions.`*out_name*`.`*domain*`.dat`, where *out_name* is the user-defined name for an HMC output location and *domain* is the domain to which the fractions belong. Output files, which report the fraction of flow into an output location from various sources, are comprised of the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Q_total` | [$L^3$ $T^{-1}$] | Total flow rate into an output location. |
| `f_reset` | [$L^3$ $T^{-1}$] | Fraction of flow associated with HMC cells that were reset due to numerical instability of the HMC algorithm. |
| `f_`*zone_name*`_initial` | [$L^3$ $T^{-1}$] | Fraction of flow associated with initial water in a tagged HMC zone. |
| `f_`*zone_name_bc_name* | [$L^3$ $T^{-1}$] | Fraction of flow associated with a boundary condition source that entered the model through a tagged HMC zone. |
| `Q_error` | [$L^3$ $T^{-1}$] | Water balance absolute error, defined as `Q_total` minus the sum of all reported flow fractions. |
| `%Q_error` | [-] | Percent water balance error, defined as $100 \cdot Q_{\text{error}}/Q_{\text{total}}$. |

Note that the variables `f_`*zone_name*_initial and `f_`*zone_name_bc_name* are dynamically named, where *zone_name* is a user-defined name for an HMC zone and *bc_name* is the name of a compatible boundary condition for an HMC zone. For each additional HMC zone (and boundary conditions acting on those zones) there will be additional variables listed in the output files. For each domain, if there exist nodes that do not belong to any HMC zone, then fractions named `f_default_`*domain_zone_name_bc_name* are created automatically.

The following commands may be used to activate and control HMC flow tracking.

## HMC zone from chosen nodes

***This command is not available with all versions of HydroGeoSphere.***

***Contact sales@aquanty.com for more information on this functionality.***

1. **zone_name** Descriptive name for HMC zone, up to 40 characters.

2. **persistent_fraction** Logical value (T/F), which if true, causes the zone to consist of a single fixed fraction that persists for the duration of the simulation. Any water entering this zone will be assigned the fraction of this zone (overwriting any previous fraction information).

Defines a new HMC zone from chosen nodes within the currently active model domain (see Section 2.8.1). Water that is initially associated with these nodes (i.e., initial conditions) is tagged as a distinct HMC source. Furthermore, water entering the defined zone through a compatible boundary condition is tagged as a distinct HMC source. Note that multiple HMC zones within the same domain are not permitted to have overlapping node sets.

•••

## HMC zone bc list from chosen nodes

***This command is not available with all versions of HydroGeoSphere.***
***Contact sales@aquanty.com for more information on this functionality.***

1. **zone_name** Descriptive name for HMC zone, up to 40 characters.

2. **bc_name(i)...end** Boundary condition name list, up to 40 characters per name.

Defines a new HMC zone from chosen nodes within the currently active model domain (see Section 2.8.1). Water that is initially associated with these nodes (i.e., initial conditions) is tagged as a distinct HMC source. Furthermore, water entering the defined zone through a compatible boundary condition that belongs to the specified list of boundary conditions is tagged as a distinct HMC source. Note that multiple HMC zones within the same domain are not permitted to have overlapping node sets.

•••

## HMC output from chosen nodes

***This command is not available with all versions of HydroGeoSphere.***
***Contact sales@aquanty.com for more information on this functionality.***

1. **out_name** Descriptive name for HMC output location, up to 40 characters.

Defines a new HMC flow tracking location from chosen nodes. Detailed HMC flow tracking information is recorded at each timestep and is written to the Tecplot ASCII output files *prefix*`o.hmc_fractions.`**out_name**`.`*domain*`.dat`, where *domain* ranges over the active HMC domains (porous medium, surface, or channel).

●●●

## HMC initial fractions from output files

***This command is not available with all versions of HydroGeoSphere.***
***Contact sales@aquanty.com for more information on this functionality.***

1. **filename** File path to an HMC fraction binary output file, e.g., *prefix*`o.hmc_pm.default_pm_initial.0001`.

Assigns initial fraction values for all domains from existing HMC fraction binary output files whose file path and time suffix match those of the input filename. All such output files must have the filename format: *prefix*`o.hmc`*_domain.frac_name.suffix*, where *domain* specifies the HGS domain (*pm*, *dual*, *olf*, *frac*, *well*, *tile*, *chan*), *frac_name* is the name of an HMC fraction, and *suffix* is a zero-padded integer identifying the output file.

●●●

## HMC use downstream fractions

***This command is not available with all versions of HydroGeoSphere.***
***Contact sales@aquanty.com for more information on this functionality.***

For models that define flux nodal from outlet boundary conditions, the default behavior is for fractions of the upstream (outlet) nodes to be preserved at the downstream (inlet) nodes when updating the volume of water flowing into the downstream nodes. This command overrides the default behavior by preserving the fractions at the downstream nodes. If no flux nodal from outlet boundary conditions are defined, then this command has no effect.

●●●

### 2.10.1 HMC Run-Time Debug

The Hydraulic Mixing Cell algorithm relies on a number of parameters including several error tolerances that cannot be set directly in **grok**. Instead, these parameters, which are initialized to default values, may be set via the `hmc_debug.control` file. This file is generated automatically by **HydroGeoSphere** with the default parameter values if it does not already exist in the simulation folder. It has the following format:

```
! Read hmc_debug file (T/F)
F
! HMC error cutoff
1.00000000000000
! HMC ratio cutoff
10000.00000000000
```

```
! HMC frac reset on (T/F)
T
! HMC Modified Mixing Cell (T/F)
T
! HMC Simple Mixing Cell (T/F)
F
! HMC relative error volume
1.000000000000000E+020
! HMC minimum volume
9.999999999999999E-021
! HMC saturation cutoff
0.990000000000000
! HMC large bc flux override (T/F)
F
! HMC large bc flux override ratio
0.990000000000000
! HMC outflow only flux override (T/F)
T
! HMC outflow only flux override ratio
0.990000000000000
! HMC output fractions only (T/F)
F
! HMC output internal neighbours flux (T/F)
T
! HMC output boundary conditions flux (T/F)
F
! HMC output domain exchange flux (T/F)
F
```

Figure 2.19: HMC run-time debug file with default parameter values.

Lines that start with '!' are comments and are ignored when reading the file. The HMC algorithm requires this exact format when parsing the HMC debug file and any deviation thereof may result in an error. The first non-comment line controls whether the rest of the file is read and the parameter values therein are applied. By default, this parameter is set to false (F), but may be changed to true (T) by the user at any time during a simulation. The file will continue to be read and the parameter values updated so long as the first non-comment line is set to true.

Of particular importance are the "HMC error cutoff", "HMC ratio cutoff", "HMC relative error volume", and "HMC minimum volume" parameters, which we denote by $\tau_{\text{err}}$ [-], $\tau_{\text{ratio}}$ [-], $\tau_{\text{rel}}$ [-], and $\tau_{\text{vol}}$ [L$^3$], respectively. In order to describe these parameters we introduce the following variables. Let $\Delta t$ denote the current HGS timestep, let $S_{\text{curr},i}$ denote the storage at HMC cell $i$ at the current simulation time $(t + \Delta t)$, let $S_{\text{prev},i}$ denote the storage at HMC cell $i$ at the previous simulation time $(t)$, let $\epsilon_i$ denote the fluid mass balance error at HMC cell $i$, and let $f_{ik}$ denote the $k$th fraction at HMC cell $i$. Then after each **HydroGeoSphere** timestep the HMC algorithm checks that the following error conditions are satisfied.

**HMC error balance test:**

$$\left|1 - \sum_k f_{ik}\right| < \tau_{\text{err}}$$

The quantity inside the absolute value is referred to as the *HMC fraction error balance*. It is reported in the `hmc_error` binary output file for each HMC domain (see Appendix A).

**HMC ratio test:**

$$\max\left(\frac{|V_{\text{in},i}|}{\bar{f}_i \cdot S_{\text{curr},i}}, \frac{|V_{\text{out},i}|}{\bar{f}_i \cdot S_{\text{prev},i}}\right) < \tau_{\text{ratio}}, \quad \bar{f}_i = \sum_k f_{ik}\,,$$

where $V_{\text{in},i}$ is the volume of water entering HMC cell $i$ and $V_{\text{out},i}$ is the volume of water leaving HMC cell $i$. The quantity defined by the max function is referred to as the *HMC mixing ratio*. It is reported in the `mixing_ratio` binary output file for each HMC domain (see Appendix A).

**HMC relative error test:**

$$|\epsilon_i \cdot \Delta t| < \tau_{\text{rel}} \cdot S_{\text{curr},i}$$

**HMC minimum volume test:**

$$S_{\text{curr},i} < \tau_{\text{vol}}$$

If any one of these conditions is not met, then the fraction for the $i$th HMC cell is reset, in which case

$$f_{ik} = \begin{cases} 1, & k = 1 \\ 0, & k > 1 \end{cases}\,,$$

where $f_{i1}$ is the default reset fraction for the $i$th HMC cell. At each timestep during a simulation the HMC algorithm writes information to the console and to the *prefix*`o.lst` file. If you encounter the error message `Relative error criteria not met!!!`, then the HMC relative error test failed for at least one HMC cell. Similarly, if you encounter the error message `Maximum HMC ratio exceeded!!!`, then the HMC ratio test failed for at least one HMC cell. If you encounter the error message `Minimum volume criteria not met!!!`, then the minimum volume test failed for at least one HMC cell. In each case, you may consider increasing the corresponding tolerance in the HMC debug file.

Other noteworthy parameters include the following:

**HMC frac reset on:** If set to false (F), then resetting of fractions is disabled. Note, however, that disabling fraction resets may have an adverse effect on HMC algorithm stability, performance, and solution accuracy.

**HMC output fractions only:** If set to true (T), then all fractions reported in the `hmc_fractions` output file are scaled by $Q_{\text{total}}$.

**HMC output internal neighbours flux:** If set to true (T), then fractions reported in the `hmc_fractions` output file contain flow contributions via nodal connections with neighboring nodes outside of the output location.

**HMC output boundary conditions flux:** If set to true (T), then fractions reported in the `hmc_fractions` output file contain contributions from boundary condition fluxes.

**HMC output exchange flux:** If set to true (T), then fractions reported in the `hmc_fractions` output file contain contributions from domain exchange fluxes.

## 2.11 Particle Tracing

Particle tracing tracks the position of an idealized massless particle as it moves through the subsurface domain following the flow field until it either exits the model via a boundary condition or exits to the surface. Each particle is released from an initial location within the subsurface domain at an initial release time. Particle locations are then updated after each timestep of the simulation and are reported over a set of user specified output times. The particle initialization data can be provided using Initial particle location from file, Initial particle location by layer from file, and Initial particle location by groundwater table from file instructions. The particle initialization instructions can be used multiple times to read particles data from different files. The total count of non-empty, non-blank lines, excluding those beginning with an exclamation point (!), across all defined initialization files determines the simulation's particle count. Particle tracing is available for transient or steady-state flow simulations and can also be used in conjunction with the command Defined flow. Particle tracing generates the following output files:

- Particle trace files are Tecplot files that record the trace path for each particle from its initial release time up to the current trace output time. Along each trace path a particle's $xyz$-position [L], group ID, trace time [T], and average velocity [L T$^{-1}$] over a trace interval (consecutive trace times) are recorded. Particle trace files are named *prefix*`o.particle_trace.`*index*, where the optional *index* is the trace index, e.g., 0001, 0002, 0003, . . . , that identifies the trace output time (see the command Output times for particle locations). The file suffix for these files is *.dat* for ASCII files and is *.szplt* for binary files (SZL binary format).
- Particle travel time file, named *prefix*`o.particle_travel_time.csv`, is a CSV file that records the status, exit type, exit name, travel time [T], and travel length [L] of each particle. If a particle exits the domain via a boundary condition, then exit type is the type of boundary condition, e.g., "Flux nodal", and exit name is the user specified name of that boundary condition. If a particle exits to the surface, then the exit type and exit name are both "Overland flow". Otherwise, the exit type and exit name are both "No exit". Particle status is described in detail below.

- Particle location file, named *prefix*`o.particle_location.dat`, is a Tecplot ASCII file that records the location, group ID, and status of each particle for each trace output time. Particle status is described in detail below. The following table provides the numerical code for each reported particle status:

| Code | Status |
|---|---|
| 0 | Moving |
| 1 | Normal exit |
| 2 | Unreleased |
| 3 | Max trace time |
| 4 | Max trace count |
| 5 | Abnormal exit |
| 6 | Bad intersection |

The following status names may be reported in the particle travel time file:

- *Moving*: The particle is moving through the model domain.
- *Unreleased*: The particle has not been released yet. Each particle is initially assigned this status, which is updated to *Moving* once the simulation time exceeds a particle's release time. Unreleased particles are not tracked, however, they are written to the output files.
- *Normal exit*: The particle has exited the model domain via either a boundary condition or to the surface.
- *Max trace time*: The maximum trace time for the particle was reached (see command Maximum trace time).
- *Max trace count*: The maximum trace count for the particle was reached (see command Maximum trace count).
- *Abnormal exit*: This status is triggered for a particle when its maximum reflection count (see command Maximum particle reflection count) is continually exceeded, hence it is unable to make any progress, over a series of trace steps. It may indicate an issue with the flow field within a local region of the model domain that contains the particle.
- *Bad intersection*: An error has occurred when computing the intersection point between the particle's trajectory and the face of the current element that contains the particle. This status is uncommon and indicates a breakdown in the particle tracing numerics, typically resulting from an ill-conditioned point-plane intersection problem, for example, when the norm of the elemental velocity vector and the norm of a particle's position vector differ by many orders of magnitude. It may also arise when a large number of reflections are performed, resulting in an accumulation of round-off error.

The following commands may be used to control particle tracing in **HydroGeoSphere**. Note that if a command is issued more than once, then the most recent invocation of that command will be honoured.

## Trace particle

Causes **HydroGeoSphere** to do particle tracing.

•••

## Trace particle logging

Enables particle tracing logging, which writes detailed information for each particle to the ASCII file *prefix*`o.trace_particle.log`. This command is intended primarily for debugging purposes and should be used with a small number of particles to keep the log file from ballooning in size.

•••

## Initial particle location from file

1. **filename** Filename of the initial particle location file.

Specifies the initial location and release time for each particle. Each line of the space-delimited input file specifies the $xyz$-coordinates [L] of the initial location, group ID, and the release time [T]. The group ID is an integer number used for Tecplot plotting purposes in the output files; if you are unsure how to specify it, simply set it to 1 for all particles. An example of a valid input file for five particles is as follows:

```
466244.6016  6355628.938  314.5798094  1  0.0
465991.6488  6356609.130  319.2335455  1  0.0
465817.7438  6357510.274  325.8648359  1  0.0
468331.4620  6355233.699  326.3972622  1  0.0
469074.5108  6355123.032  316.3109862  1  0.0
```

Note that if a particle's initial location is outside the model domain, then its initial location is set to the centroid of the nearest element.

•••

## Initial particle location by layer from file

1. **filename** Filename of the initial particle location file.

Specifies the initial location and release time for each particle. Each line of the space-delimited input file specifies the $xy$-coordinates [L] and layer number of the initial location, group ID, and the release time [T]. The group ID is an integer number used for Tecplot plotting purposes in the output files; if you are unsure how to specify it, simply set it to 1 for all particles. Note that when prescribing a particle's initial location by layer, its $z$-coordinate is defined as $(z_{\text{top}} + z_{\text{bot}})/2$, where the points $(x, y, z_{\text{top}})$ and $(x, y, z_{\text{bot}})$ belong to the top and bottom faces, respectively, of the element that contains the particle. An example of a valid input file for five particles all belonging to layer three is as follows:

```
466244.6016  6355628.938  3  1  0.0
465991.6488  6356609.130  3  1  0.0
465817.7438  6357510.274  3  1  0.0
468331.4620  6355233.699  3  1  0.0
469074.5108  6355123.032  3  1  0.0
```

Note that if a particle's initial location is outside the model domain, then its initial location is set to the centroid of the nearest element.

•••

## Initial particle location by groundwater table from file

1. **filename** Filename of the initial particle location file.

Specifies the initial location and release time for each particle. Each line of the space-delimited input file specifies the $xy$-coordinates [L] of the initial location, group ID, and the release time [T]. The group ID is an integer number used for Tecplot plotting purposes in the output files; if you are unsure how to specify it, simply set it to 1 for all particles. When prescribing a particle's initial location by groundwater table, its $z$-coordinate is interpolated to the initial groundwater table elevation at the point $(x, y)$. Note that for a fully saturated model, the $z$-coordinate of each particle is interpolated to the top node sheet. An example of a valid input file for five particles is as follows:

```
466244.6016  6355628.938  1  0.0
465991.6488  6356609.130  1  0.0
465817.7438  6357510.274  1  0.0
468331.4620  6355233.699  1  0.0
469074.5108  6355123.032  1  0.0
```

Note that if a particle's initial location is outside the model domain, then its initial location is set to the centroid of the nearest element.

•••

## Output times for particle locations

1. **time(i)...end** Output time [T] list.

Specifies the output times at which particle trace files are generated/updated and the times at which particle locations are recorded in the particle location file.

• • •

## No particle trace files

Disables the generation of particle trace files (particle location and travel time files are still generated). By default, particle trace files are generated/updated for each output time specified by the command Output times for particle locations.

• • •

## Particle trace binary output

1. **single_file** Logical value (T/F), which if true causes all particle traces to be written to a single file named *prefix*`o.particle_trace.szplt`. Otherwise, a new particle trace output file is generated for each output time.

Causes particle trace files to be formatted as Tecplot binary files (SZL binary format) with the *.szplt* file suffix. Note that by default particle trace files are formatted as Tecplot ASCII files with one file per output time.

• • •

## Particle trace timestep threshold

1. **threshold** Timestep threshold [T].

Specifies a threshold for particle timesteps that controls whether a new particle location is recorded or whether the current particle location is updated. A new particle location is recorded when the trace time since the last update exceeds **threshold**. This command has a smoothing effect on particle trace paths with the goal of reducing memory usage and `o.particle_trace` file size. By default the timestep threshold is set to zero, which corresponds to each new location along a particle trace path being recorded. Note that the quantities “travel length” and “average velocity” reported in the output files will be defined in terms of the smoothed trace paths.

• • •

## Maximum trace time

1. **max_time** Maximum trace time [T].

Specifies the maximum time at which particle traces are updated. For each particle, tracing effectively stops after this time. By default the maximum trace time is effectively unlimited.

● ● ●

## Maximum trace count

1. **max_count** Maximum trace count.

Specifies the maximum number of locations in a particle trace path. This value can be used to control the amount of memory consumed by particle tracing. For each particle, tracing effectively stops once this count is reached. By default the maximum trace count is effectively unlimited, in which case the main constraint on the number of locations in a particle trace path is the amount of virtual memory available.

● ● ●

## Maximum trace output

1. **max_output** Maximum number of particles to output.

Specifies the maximum number of particles to record in the particle trace and particle location output files. This value can be used to control the amount of memory consumed by particle tracing output files. Output is recorded only for particles whose IDs belong to the range $1, \ldots,$ **max_output**. Note that a particle's ID is determined by the line number of its initial location within the initial location input file. By default the maximum trace output is set to 10 000.

● ● ●

## Maximum particle reflection count

1. **max_reflect** Maximum number of particle relfections per trace step.

Specifies the maximum number of particle reflections that are permitted over a trace step when updating a particle's location. A particle will reflect during a trace step if it is unable to move from one element to a neighboring element through an adjacent face as a result of the current flow field. If this number is exceeded for a given particle, then the trace step for that particle is stopped and the particle location is not updated. Tracing for that particle then resumes at the next timestep. By default the maximum particle reflection count is set to 100.

● ● ●

## 2.12 Output

During execution, **grok** and **HydroGeoSphere** create many output files, for which a complete list with brief descriptions can be found in Appendix A. Here, we will discuss in more detail the output that is of most interest to the user.

The main mechanism for generating output from **HydroGeoSphere** is through the use of the Output times instruction. For each output time defined by the user, there will be a problem dependent set of output files that are generated automatically. Here is a partial list of the files that were created by the verification problem described in Section 1.5.2 of the verification manual:

```
f_cdo.conc_frac.Radium.0001
f_cdo.conc_frac.Thorium.0001
f_cdo.conc_frac.Uranium.0001
f_cdo.conc_pm.Radium.0001
f_cdo.conc_pm.Thorium.0001
f_cdo.conc_pm.Uranium.0001
f_cdo.head_frac.0001
f_cdo.head_pm.0001
f_cdo.q_pm.0001
f_cdo.v_frac.0001
f_cdo.v_pm.0001
```

As you can see, file names are made up of the problem prefix, in this case `f_cd`, a descriptor, for example, `o.conc_frac.Radium`, and a 4-digit number such as 0001, which relates the data contained in the file to an output time number. Note that for output time number 0001, there are quite a few more output files than for number 0002. That is because the flow solution in this case is steady-state, and so it is fully described by one set of outputs. For time number 0002, only variables that have changed are output, in this case the concentrations of the three species in the transport solution. The set of files that are generated is problem dependent, hence, certain types of files may or may not appear. For example, since this problem has discrete fractures, we are seeing some fracture velocity output.

In order to conserve disk space and increase the efficiency of file I/O, these files are all stored in binary format, so they can not be viewed with a standard text editor. However, through the use of the post-processing tools **HSPLOT** (Appendix D) or **HGS2VTU** (Appendix E), it is possible to create ASCII or binary versions of the data contained in these files that are compatible with the third-party visualization tools such as Tecplot.

Users are granted fine-grained control over which binary output files are generated by **HydroGeoSphere** via the *prefix*`o.output_variable.control` file. This file is created automatically by **HydroGeoSphere** unless one is already present in the working directory. It contains a list of file descriptors followed by either “yes” or “no”, where “yes” causes the corresponding output file to be generated and “no” suppresses its output. Note that if the output variable control file becomes corrupted or out of date you can automatically generate a fresh copy by deleting or renaming it before running **HydroGeoSphere**.

By default, simulation time is written in listing and output files in a scientific format using the Fortran `E` format descriptor (`1PE17.10`). If you prefer to use fixed format notation (`F17.5`), which is suitable for times between 0.00001 and 999 999, then you can do so via the command:

```
Time output fixed format
```

Another option is to use the Fortran `G` format descriptor (i.e., general format) in which a mix of fixed and scientific format is used depending on the magnitude of the output. If you prefer this notation, then you can choose it via the command:

```
Time output general format
```

which uses the Fortran format descriptor `1PG17.10`.

Since similar versions of these commands can be used in the `debug.control` file (see Appendix B), it may also be useful to switch back to scientific format. To do so, use the command:

```
Time output scientific format
```

Similarly, mass balance output formatting is controlled by the commands:

```
Mass balance output scientific format
Mass balance output general format
Mass balance output fixed format
```

which use the Fortran format descriptors `1PE20.10`, `1PG20.10`, and `F20.10`, respectively. By default, mass balance output formatting uses scientific notation.

For simulations with a large number of output times, the default 4-digit output time number may not be able to index all output times. In that case, you can use the following command to increase the number of digits in the output time number.

## Number of digits for output file suffix

1. **suffix_len** Number of digits in the output file suffix, an integer between 4 and 10, inclusive.

Sets the number of digits in the output file suffix to **suffix_len**. By default the number of digits is set to four. For example, setting the number of digits to five would result in output time numbers: 00001, 00002, . . . , enabling you to index up to 99 999 output times.

● ● ●

### 2.12.1 Mesh Quality

Creating a high-quality mesh is one of the most important factors that should be considered to ensure simulation accuracy, convergence, and performance for any mesh-based numerical model. To assess mesh quality, **grok** provides summary statistics for several pertinent metrics as described in Table 2.27 (see Stimpson et al. (2007) for details). The column labeled "Acceptable Range" is meant to provide guidance on what constitutes a "good" value for each metric. Metrics are computed for each element in the 2-D mesh that is obtained by projecting the 3-D mesh onto the *xy*-plane, and summary statistics are written to the *prefix*`o.eco` file after the mesh summary section.

Table 2.27: Mesh quality metrics.

| Metric | Units | Description | Acceptable Range |
|---|---|---|---|
| Area | [L$^2$] | Element area | |
| Aspect Ratio | [-] | Measure of element quality | $[1, 2]$ |
| Edge Ratio | [-] | Ratio of max to min edge lengths | $[1, 2]$ |
| Minimum Angle | [deg] | Minimum interior angle | [30°, 60°] |
| Maximum Angle | [deg] | Maximum interior angle | [60°, 90°] |
| Right Triangle Distance | [deg] | Measure of nearness to right triangle | $> 2°$ |

In addition to the summary statistics reported in the *prefix*`o.eco` file, you can use the command Report mesh quality to export the mesh quality metrics to a Tecplot ASCII file for the entire 2-D mesh.

#### Report mesh quality

Causes **grok** to report mesh quality metrics for each element in the 2-D mesh that is obtained by projecting the 3-D mesh onto the *xy*-plane. Output is written to the Tecplot ASCII file *prefix*`o.mesh_quality.dat`. Note that the metric "Right Triangle Distance" in Table 2.27 is reported only for triangular meshes.

• • •

### 2.12.2 Grid

The following instructions can be useful in checking the results of the grid generation section.

#### Echo coordinates

Causes node coordinates to be written to the *prefix*`o.eco` file.

• • •

#### Echo incidences

Causes element incidences to be written to the *prefix*`o.eco` file.

●●●

### Echo fracture incidences

Causes fracture element incidences to be written to the *prefix*`o.eco` file.

●●●

### List surface flow nodes

Causes the list of surface flow nodes to be written to the *prefix*`o.eco` file.

●●●

### Write faces and segments

Whenever **HydroGeoSphere** generates a 3-D mesh, it makes lists of the nodes which comprise each unique face and line segment. This information is used by certain instructions which choose faces or segments. You can use this instruction to write the information to a file which will receive the name *prefix*`o.fac`. These files can become quite large, so the default is not to save them. See also Section 2.3.6.

●●●

### Elevation to file

1. **filename** Filename of the elevations ASCII output file.
2. **sheet** Node sheet number for which to write elevations.

Writes the nodal elevations [L] for the specified 2-D node sheet to the specified file. The order of the elevations is the same as the node order within a 2-D node sheet. The file generated by this command can be used to specify 2-D node sheet elevations during grid generation via the command Elevation from file. Note that the 2-D mesh topology must remain unchanged in order to use this command in conjunction with the Elevation from file command.

●●●

#### 2.12.2.1 GMS

The following instructions can be used to export data generated by **grok** (e.g., 2-D and 3-D meshes, etc.) to GMS.

### Mesh to gms

1. **gmsfile** Name of the GMS mesh file.

Writes 3-D mesh information (blocks or prisms) in GMS-readable format to the specified file.

• • •

### 2D mesh to gms

1. **gmsfile** Name of the GMS mesh file.

Writes 2-D mesh information (quadrilaterals or triangles) in GMS-readable format to the specified file.

• • •

### Rectangles to triangles

1. **gmsfile** Name of the GMS mesh file.

Writes 2-D triangular mesh information in GMS-readable 2-D mesh format to the specified file. Each rectangle in the 2-D mesh is split into two triangles. Nodal coordinates are written first followed by element node lists.

• • •

## 2.12.3 Tecplot

The following instructions can be used to export data generated by **grok** to Tecplot.

### Mesh to tecplot

1. **tecfile** Name of the file to write 3-D mesh information.

Writes 3-D mesh information to a Tecplot ASCII file with the user specified name.

• • •

### Overland to tecplot

1. **tecfile** Name of the file to write the overland domain mesh information.

Writes overland domain mesh information to a Tecplot ASCII file with the user specified name.

•••

## Fractures to tecplot

1. **tecfile** Name of the file to write the fracture domain mesh information.

Writes fracture domain mesh information to a Tecplot ASCII file with the user specified name.

•••

## Channels to tecplot

1. **tecfile** Name of the file to write the channel domain mesh information.

Writes channel domain mesh information to a Tecplot ASCII file with the user specified name.

•••

## Wells to tecplot

1. **tecfile** Name of the file to write the well domain mesh information.

Writes well domain mesh information to a Tecplot ASCII file with the user specified name.

•••

## Tiles to tecplot

1. **tecfile** Name of the file to write the tile domain mesh information.

Writes tile domain mesh information to a Tecplot ASCII file with the user specified name.

•••

## K to tecplot

1. **tecfile** Name of the file to write hydraulic conductivity.

Writes hydraulic conductivity information to a Tecplot ASCII file with the user specified name.

●●●

## Porosity to tecplot

1. **tecfile** Name of the file to write porosity.

Writes porosity information to a Tecplot ASCII file with the user specified name.

●●●

## Tortuosity to tecplot

1. **tecfile** Name of the file to write tortuosity.

Writes tortuosity information to a Tecplot ASCII file with the user specified name.

●●●

## ET zones to tecplot

1. **tecfile** Name of file to write ET zones.

Writes ET zone information to a Tecplot ASCII file with the user specified name.

●●●

## Chosen nodes to tecplot

1. **tecfile** Name of file to write node selection.

Writes the current node selection to a Tecplot ASCII file with the user specified name.

●●●

## Chosen elements to tecplot

1. **tecfile** Name of file to write element selection.

Writes the current element selection to a Tecplot ASCII file with the user specified name.

●●●

### 2.12.4 Flow Solution

The following instructions affect the I/O for the flow solution.

---

#### Echo to output

Causes heads, saturations, concentrations, and velocities for the subsurface domain to be written to the *prefix*`o.lst` file as well as to the binary output files.

●●●

---

#### Track simulation progress

Causes **HydroGeoSphere** to generate a CSV output file named *prefix*`o.simulation_progress.csv` that tracks the progress of an HGS simulation. The file, which is overwritten after each successful timestep, reports the following variables:

- `sim time:` Simulation time [T] at the start of each timestep.
- `%done:` Percentage of the simulation completed as measured from the initial start time to the reported simulation time. Note that the final target time is used as the end time of the simulation.
- `wall time:` Simulation wall clock time in seconds.
- `cpu time:` Simulation CPU time in seconds.

●●●

---

#### Generate time report

Causes **HydroGeoSphere** to write a Tecplot ASCII output file *prefix*`o.sim_time_report` with the information in the `SIMULATION TIME REPORT` found at the end of the *prefix*`o.lst` file. The meaning of each heading in the output file is described by the following table:

| File Heading | Simulation Time Report |
|---|---|
| CPU | Number of CPUs applied |
| NEQ | Number of equations |
| Time_step | Number of time steps |
| NR_iter | Number of total Newton-Raphson loops |
| Solver_iter | Number of total solver iterations |
| Global_assembly | Global assembly time (s) |
| Precondition | ILUC time (s) |
| Solver | Solver time (s) |
| BC_sets | SetBC time (s) |

| | |
|---|---|
| 4_Major_times | Sum of four previous times |
| Total_time | Simulation wall time (s) |

●●●

## Flux output nodes

1. **new_noutfc** Number of new output nodes desired.
2. **ioutfc(i), i=1,new_noutfc** Flux output node number.

Listed nodes are flagged as flux output nodes, at which detailed fluid flux [$L^3$ $T^{-1}$] information for the subsurface domain is output at each timestep. Flux output is written to a file called *prefix*`o.flu`. For each timestep in the flow solution, one line per flux output node will be written to the file. Each line contains the node number, time, fluid flux and nodal coordinates. Such output can be imported into an editor (e.g., Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of fluid flux versus time at a node.

●●●

## Flux output nodes from chosen

This instruction works in a similar way to Flux output nodes except that all currently chosen nodes are flagged as flux output nodes.

●●●

## Binary flux output nodes

1. **new_noutfc** Number of new output nodes desired.
2. **ioutfc(i), i=1,new_noutfc** Flux output node number.

This instruction works in a similar way to Flux output nodes, however, output is written to a binary file named *prefix*`o.flu_b`.

●●●

## Binary flux output nodes from chosen

This instruction works in a similar way to Binary flux output nodes except that all currently chosen nodes are flagged as flux output nodes.

●●●

## Output saltwater head

By default, when running density-dependent flow problems, the equivalent freshwater head is written to the head file *prefix*`o.head_pm.`*suffix*, where suffix is a zero-padded integer identifying the output file. This instruction causes salt water heads to be written instead.

●●●

## Output ET details

This command reports detailed ET information for each Potential evapotranspiration boundary condition in the file *prefix*`o.ET_Detailed_MB_`*bcname*`.dat`, where *bcname* is the named of the PET boundary condition.

●●●

#### 2.12.4.1 Observation Wells and Points

Observation well/point commands output timeseries information at a single node or a group of nodes lying on a line. Output is written to a Tecplot ASCII file named *prefix*-`o.observation_well_flow.`*well_name*`.dat`, where *well_name* is a user specified identifier for an observation well/point. Depending on model setup, for example, the domains/processes defined, the following variables may be reported in the output file:

| Variable | Units | Domain | Description |
|---|---|---|---|
| `Time` | [T] | | Simulation time (transient) |
| `H` | [L] | PM | Total head |
| `P` | [L] | PM | Pressure head |
| `S` | [$L^3$ $L^{-3}$] | PM | Water saturation (variably saturated) |
| `VWC` | [$L^3$ $L^{-3}$] | PM | Volumetric water content (variably saturated) |
| `Q` | [$L^3$ $T^{-1}$] | PM | Nodal fluid flux |
| `T` | [°C] | PM | Temperature (transient, freezing-thawing) |
| `I` | [$L^3$ $L^{-3}$] | PM | Ice saturation (transient, freezing-thawing) |
| `Hd` | [L] | Dual | Total head |
| `Pd` | [L] | Dual | Pressure head |
| `Sd` | [$L^3$ $L^{-3}$] | Dual | Water saturation (variably saturated) |
| `Qd` | [$L^3$ $T^{-1}$] | Dual | Nodal fluid flux |
| `Id` | [$L^3$ $L^{-3}$] | Dual | Ice saturation (transient, freezing-thawing) |
| `Ho` | [L] | Overland | Total head |

| | | | |
|---|---|---|---|
| `Po` | [L] | Overland | Pressure head |
| `Qo` | [$L^3$ $T^{-1}$] | Overland | Nodal fluid flux |
| `Hf` | [L] | Fracture | Total head |
| `Pf` | [L] | Fracture | Pressure head |
| `Qf` | [$L^3$ $T^{-1}$] | Fracture | Nodal fluid flux |
| `Hw` | [L] | Well | Total head |
| `Qw` | [$L^3$ $T^{-1}$] | Well | Nodal fluid flux |
| `Ht` | [L] | Tile | Total head |
| `Qt` | [$L^3$ $T^{-1}$] | Tile | Nodal fluid flux |
| `Hc` | [L] | Channel | Total head |
| `iHc` | [L] | Channel | Incised total head |
| `Qc` | [$L^3$ $T^{-1}$] | Channel | Nodal fluid flux |
| `T_PM` | [$L^3$ $T^{-1}$] | ET | Subsurface transpiration (transient) |
| `E_PM` | [$L^3$ $T^{-1}$] | ET | Subsurface evaporation (transient) |
| `Compaction` | [L] | PM | Subsurface compaction |

In addition, the $xyz$-coordinates, surface elevation at $(x, y)$, and node number are reported for each node that forms the observation well/point. Note that if a node does not belong to a given domain, then the no data value ($-999.0$) is reported.

The following commands may be used to define an observation point.

## Make observation point

1. **name** Descriptive name for the observation point, up to 40 characters.
2. **x, y, z** $xyz$-coordinates [L] of the observation point.

The node closest to the input point is defined as an observation point.

●●●

## Make observation points

1. **name(i), x(i), y(i), z(i)...end** Descriptive name (up to 40 characters, no spaces) and $xyz$-coordinates [L] of the observation point.

For each point in the input list, the node closest to that point is defined as an observation point.

●●●

## Make observation points from shp

1. **filename** Name of the shapefile without the file extension.
2. **name_attribute** Field name (up to 11 characters) used to assign the observation point name, which must be written exactly as in the `.dbf` file (case sensitive).
3. **z_attribute** Field name (up to 11 characters) used to assign the observation point $z$-coordinate [L], which must be written exactly as in the `.dbf` file (case sensitive).

For each point in the shapefile, the node closest to that point is defined as an observation point. Point, PointM, and PointZ ($z$-coordinate ignored) shape types are supported. The name and $z$-coordinate of each observation point is read from the corresponding database file (`.dbf`).

●●●

## Make interpolated observation point

1. **name** Descriptive name for the observation point, up to 40 characters.
2. **x, y, z** $xyz$-coordinates [L] of the observation point.

Output values are linearly interpolated from the nodal values at the vertices of the element that contains the observation point. Note that this command applies only to the porous media domain.

●●●

## Make interpolated observation points

1. **name(i), x(i), y(i), z(i)...end** Descriptive name (up to 40 characters, no spaces) and $xyz$-coordinates [L] of the observation point.

For each point in the input list, output values are linearly interpolated from the nodal values at the vertices of the element that contains that observation point. Note that this command applies only to the porous media domain.

●●●

## Make interpolated observation points by layer from shp

1. **filename** Name of the shapefile without the file extension.
2. **name_attribute** Field name (up to 11 characters) used to assign the observation point name, which must be written exactly as in the `.dbf` file (case sensitive).

3. **layer_attribute** Field name (up to 11 characters) used to assign the observation point layer number, which must be written exactly as in the `.dbf` file (case sensitive).

For each point in the shapefile, output values are linearly interpolated from the nodal values at the vertices of the element that contains that observation point. Point, PointM, and PointZ ($z$-coordinate ignored) shape types are supported. The name and layer number of each observation point is read from the corresponding database file (`.dbf`). For each coordinate pair $(x, y)$ that defines an observation point, its $z$-coordinate is defined as $(z_{\mathrm{top}} + z_{\mathrm{bot}})/2$, where the points $(x, y, z_{\mathrm{top}})$ and $(x, y, z_{\mathrm{bot}})$ belong to the top and bottom faces, respectively, of theelement in the specified mesh layer that contains the point $(x, y)$.

• • •

## Make observation point by sheet

1. **name** Descriptive name for the observation point, up to 40 characters.
2. **x, y, sheet** $xy$-coordinates [L] and sheet number of the observation point.

The node closest to the input point in the specified node sheet is defined as an observation point.

• • •

## Make observation points by sheet

1. **name(i), x(i), y(i), sheet(i)...end** Descriptive name (up to 40 characters, no spaces), $xy$-coordinates [L], and sheet number of the observation point.

For each point in the input list, the node closest to that point in the specified node sheet is defined as an observation point.

• • •

## Make observation point by depth

1. **name** Descriptive name for the observation point, up to 40 characters.
2. **x, y, depth** $xy$-coordinates [L] and depth [L] of the observation point.

The node closest to the input point is defined as an observation point.

• • •

## Make observation points by depth

1. **name(i), x(i), y(i), depth(i)...end** Descriptive name (up to 40 characters, no spaces), *xy*-coordinates [L], and depth [L] of the observation point.

For each point in the input list, the node closest to that point is defined as an observation point.

●●●

## Make observation points by depth from shp

1. **filename** Name of the shapefile without the file extension.
2. **name_attribute** Field name (up to 11 characters) used to assign the observation point name, which must be written exactly as in the `.dbf` file (case sensitive).
3. **depth_attribute** Field name (up to 11 characters) used to assign the observation point depth [L], which must be written exactly as in the `.dbf` file (case sensitive).

For each point in the shapefile, the node closest to that point is defined as an observation point. Point, PointM, and PointZ ($z$-coordinate ignored) shape types are supported. The name and depth value of each observation point is read from the corresponding database file (`.dbf`).

●●●

## Make interpolated observation point by depth

1. **name** Descriptive name for the observation point, up to 40 characters.
2. **x, y, depth** *xy*-coordinates [L] and depth [L] of the observation point.

Output values are linearly interpolated from the nodal values at the vertices of the element that contains the observation point. Note that this command applies only to the porous media domain.

●●●

## Make interpolated observation points by depth

1. **name(i), x(i), y(i), depth(i)...end** Descriptive name (up to 40 characters, no spaces), *xy*-coordinates [L], and depth [L] of the observation point.

For each point in the input list, output values are linearly interpolated from the nodal values at the vertices of the element that contains that observation point. Note that this command applies only to the porous media domain.

•••

## Make node observation point

1. **name** Descriptive name for the observation point, up to 40 characters.
2. **number** Node number of the observation point.

Defines an observation point at the specified node number.

•••

## Make node observation points

1. **name(i), number(i)...end** Descriptive name (up to 40 characters, no spaces) and node number of observation point.

Defines an observation point at the specified node number for each entry in the input list.

•••

The following commands may be used to define an observation well.

## Make observation well

1. **well_name** Descriptive name for the well, up to 40 characters.
2. **x1, y1, z1** *xyz*-coordinates [L] of one end of the observation well.
3. **x2, y2, z2** *xyz*-coordinates [L] of the other end of the observation well.

Element face edges (i.e., segments) that form the shortest path between the two nodes closest to the end points of the well (i.e., (**x1**, **y1**, **z1**) and (**x2**, **y2**, **z2**)) are selected. Each timestep in the flow solution represents a Tecplot zone in the output file and each zone contains one line per observation well node.

•••

## Make observation well from xy

1. **well_name** Descriptive name for the well, up to 40 characters.
2. **x, y** *xy*-coordinates [L] of the observation well.

3. **isheet1, isheet2** Bottom and top sheet numbers (inclusive).

The *xy*-coordinates together with the sheet numbers define the end points of the well. Element face edges (i.e., segments) that form the shortest path between the two nodes closest to these well end points are then selected. Each timestep in the flow solution represents a Tecplot zone in the output file and each zone contains one line per observation well node.

●●●

#### 2.12.4.2 Fluid Mass Balance

By default, fluid mass balance information is computed at each time step and written to the *prefix*`o.lst` file. Figure 2.20 illustrates some sample output as it appears in that file.

```
--------------------------------------------------------------------
 FLUID BALANCE, TIME:   0.500000000000000
--------------------------------------------------------------------
RATE OF FLUID EXCHANGE              IN              OUT             NET(IN-OUT)
TOTAL
   Fixed flux                     0.0059395897    0.0000000000    0.0059395897
   Critical depth                                 0.0000000000    0.0000000000
   NET1 EXCHANGE RATE (IN-OUT)                                    0.0059395897

RATE OF FLUID ACCUMULATION [L**3/T]
   Porous medium                  0.0003197172
   Overland                       0.0057208226
   NET2 ACCUMULATION RATE                                         0.0060405397

FLUID BALANCE ERROR
   Absolute: (NET1-NET2)                                         -0.0001009500
   Relative: (NET1-NET2)/(abs(NET1)+abs(NET2))/2.0                0.0168529029
   Percent:  abs(NET1-NET2)/[max{NET1(+),abs(NET1(-)}]*100.0      1.6996119929

FLUID EXCHANGE BETWEEN SURFACE (DUAL NODES) AND SUBSURFACE DOMAIN [L**3/T]
  Infiltration                    0.0003219343
  Exfiltration                    0.0000000000
```

Figure 2.20: Example of fluid balance information written to file *prefix*`o.lst`.

This detailed fluid balance information is also written to a Tecplot formatted ASCII output file named *prefix*`o.water_balance.dat` so that it can be easily visualized. For each timestep in the flow solution one line is written to the file. The number of columns in the file depends on the nature of the problem. For example, for problems with no overland flow component, no overland flow data will be written to the file.

The first section of the water balance output file gives the volumetric rate [$L^3$ $T^{-1}$] at which water is entering or leaving all model domains via various types of sources or sinks. The volume of water at the $i$th timestep, $V_i$, can be computed from the rate, $r_i$, via the equation

$$V_i = r_i(t_i - t_{i-1}), \quad i \geq 1,$$

where consecutive time values can be read from the table and by default $t_0 = 0$. The section begins by listing the net water flux for each flow boundary condition. Column headings for the boundary conditions consist of their assigned names (assigned via the command Name). We note that by default the maximum number of boundary conditions listed in the water balance output file is capped at 100. This number can be changed via the following command.

## Maximum number of boundary conditions to report

1. **number** Maximum number of boundary conditions to report in the water balance output file *prefix*`o.water_balance.dat`.

By default the maximum number is set to 100.

●●●

The remaining part of this section may include the following column headings:

- **River+, River-** Rate at which water is entering or leaving the model via river nodes, as defined in Section 2.3.2.4 of the Theory Manual [$L^3$ $T^{-1}$].
- **Hydrostatic+, Hydrostatic-** Rate at which water is entering or leaving the model via hydrostatic head nodes [$L^3$ $T^{-1}$].
- **NET1 Sources/Sinks** Rate at which water entering or leaving all model domains through sources all and sinks, $\pm Q$ [$L^3$ $T^{-1}$].

The next section of the water balance output file relates to the changes in storage that occurred in the various model domains. It may include the following column headings:

- **PM** Rate of accumulation in the porous media domain [$L^3$ $T^{-1}$].
- **Overland** Rate of accumulation in the overland flow domain [$L^3$ $T^{-1}$].
- **Reservoir** Rate of accumulation for all reservoirs [$L^3$ $T^{-1}$] (see boundary conditions Reservoir and Reservoir with spillway).
- **Dual** Rate of accumulation in the dual continuum domain [$L^3$ $T^{-1}$].
- **Fracture** Rate of accumulation in discrete fractures [$L^3$ $T^{-1}$].
- **Channels** Rate of accumulation in 1-D channels [$L^3$ $T^{-1}$].
- **Tiles** Rate of accumulation in tile drains [$L^3$ $T^{-1}$].

- **Wells** Rate of accumulation in wells [L$^3$ T$^{-1}$].
- **ET Interception Storage** Rate of change of water stored as canopy interception [L$^3$ T$^{-1}$].
- **NET2 Accumulation** Rate of change of storage for all model domains $\Delta S$ [L$^3$ T$^{-1}$].

In a perfectly balanced system the rate at which water enters or leaves the domain $\pm Q$ and the rate of change in storage $\Delta S$ would be equal and the fluid balance error $\varepsilon$ would be

$$\varepsilon = \pm Q - \Delta S = 0 \tag{2.8}$$

The next section of the water balance output file relates to the fluid balance error, which can be expressed in various ways:

- **ERROR (NET1-NET2)** Absolute error $\varepsilon$ [L$^3$ T$^{-1}$].
- **Error rel** Relative error [-]:

$$\varepsilon_R = \frac{2|\varepsilon|}{|\pm Q| + |\Delta S|} \tag{2.9}$$

- **Error percent** Percent error [-]:

$$\varepsilon_P = 100 \cdot \frac{|\varepsilon|}{\max(Q_+,\, |Q_-|)}, \tag{2.10}$$

  where $\pm Q = Q_+ + Q_-$, $Q_+ \geq 0$ is the net movement into the domain, and $Q_- \leq 0$ is the net movement out of the domain.

The next section of the water balance output file gives the fluid exchange between the various domains defined in your model. Fluid exchange is reported only for those domains that use the dual node approach. By convention, the rate at which water is added to a domain is positive and the rate at which it is removed is negative, e.g., **Olf2PM** is positive and **PM2Olf** is negative. This section may include the following column headings:

- **Olf2PM, PM2Olf** Rate at which water is being added to or removed from the subsurface domain via the surface domain (dual nodes) [L$^3$ T$^{-1}$].
- **Olf2Dual, Dual2Olf** Rate at which water is being added to or removed from the dual continuum domain via the surface domain (dual nodes) [L$^3$ T$^{-1}$].
- **Dual2PM, PM2Dual** Rate at which water is being added to or removed from the subsurface domain via the dual continuum domain [L$^3$ T$^{-1}$].
- **Frac2PM, PM2Frac** Rate at which water is being added to or removed from the subsurface domain via fractures (dual nodes) [L$^3$ T$^{-1}$].

- **Well2PM, PM2Well** Rate at which water is being added to or removed from the subsurface domain via wells (dual nodes) [$L^3$ $T^{-1}$].
- **Tile2PM, PM2Tile** Rate at which water is being added to or removed from the subsurface domain via tile drains (dual nodes) [$L^3$ $T^{-1}$].
- **Chan2PM, PM2Chan** Rate at which water is being added to or removed from the subsurface domain via 1-D channels (dual nodes) [$L^3$ $T^{-1}$].
- **Chan2Olf, Olf2Chan** Rate at which water is being added to or removed from the surface domain via 1-D channels (dual nodes) [$L^3$ $T^{-1}$].

The final section of the water balance output file gives a detailed breakdown for evapotranspiration (if being used) and includes the following column headings:

- **Canopy_evap** Rate at which water is leaving canopy storage as evaporation [$L^3$ $T^{-1}$].
- **Surf_evap** Rate at which water is leaving the overland flow domain via evaporation [$L^3$ $T^{-1}$].
- **PM_evap** Rate at which water is leaving the porous media domain via evaporation [$L^3$ $T^{-1}$].
- **PM_trans** Rate at which water is leaving the porous media domain via transpiration [$L^3$ $T^{-1}$].
- **Tot_ET** Total rate at which water is leaving the model domain for all evaporation and transpiration components for both the overland flow and porous media domains [$L^3$ $T^{-1}$].

Note that depending on the model setup, the user will likely want to consult additional output files that compliment the water balance file. These files include, but are not limited to:

- *prefix*`o.Rain_Snowmelt_balance.dat`: generated when boundary condition Rain and snowmelt is used.
- *prefix*`o.reservoir_`*bcname*`.dat`: generated for each Reservoir boundary condition.
- *prefix*`o.reservoir_spillway.`*bcname*`.dat`: generated for each Reservoir with spillway boundary condition.
- *prefix*`o.snow_balance.dat`: generated when boundary condition Snowmelt is used.
- *prefix*`o.soil_balance.dat`: generated when either of the commands Soil water balance or Soil water balance custom depth are used.

#### 2.12.4.3 Soil Water Balance

## Soil water balance

This command generates timeseries information on the volume of water stored in the partially-saturated and fully-saturated subsurface, and in the surface. Output is written to the Tecplot ASCII file *prefix*`o.soil_balance.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Soil` | [$L^3$] | Volume of water stored in unsaturated zone. |
| `Groundwater` | [$L^3$] | Volume of water stored in the saturated zone. |
| `Top 0.1000 m` | [$L^3$] | Volume of water stored in the top 0.1 m in the subsurface. |
| `Top 1.000 m` | [$L^3$] | Volume of water stored in the top 1.0 m in the subsurface. |
| `Total PM` | [$L^3$] | Volume of water stored in the subsurface. |
| `Surface` | [$L^3$] | Volume of water stored in the surface. |

Note that the variables `Top 0.1000 m` and `Top 1.000 m` change based on the model units. For example, if the units of length is centimeters, then these variables would appear as `Top 10.00 cm` and `Top 100.0 cm`, respectively.

●●●

## Soil water balance custom depth

1. **depth(i)...end** Depths [L] in the subsurface.

This command generates timeseries information on the volume of water stored in the partially-saturated and fully-saturated subsurface, in the subsurface from the top of the model to the depths in the input list, and in the surface. Output is written to the Tecplot ASCII file *prefix*`o.soil_balance.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Soil` | [$L^3$] | Volume of water stored in unsaturated zone. |
| `Groundwater` | [$L^3$] | Volume of water stored in the saturated zone. |
| `Top` $d_1$ | [$L^3$] | Volume of water stored in the top $d_1$ length units in the subsurface. |
| `Top` $d_2$ | [$L^3$] | Volume of water stored in the top $d_2$ length units in the subsurface. |
| $\vdots$ | $\vdots$ | $\vdots$ |
| `Total PM` | [$L^3$] | Volume of water stored in the subsurface. |
| `Surface` | [$L^3$] | Volume of water stored in the surface. |

Note that the variable names `Top` $d_1$, `Top` $d_2, \ldots$ have the same length units as the model.

●●●

#### 2.12.4.4 Water Table

## Report water table at xy

1. **name** Descriptive name, up to 40 characters.
2. **x, y** *xy*-coordinates [L] of the location at which to compute the water table.

Water table position is computed along the vertical column that contains the node which is nearest to the input location and is reported as "depth to water table" and "water table elevation". Output is written to the Tecplot ASCII file *prefix*`o.water_table.`**name**`.dat`. For each timestep in the flow solution, one line is written to the file. Each line contains the simulation time [T], depth to water table [L], and water table elevation [L].

●●●

## Report water table at xyz

1. **name** Descriptive name, up to 40 characters.
2. **x, y, z** *xyz*-coordinates [L] of the location at which to compute the water table.

Water table position is computed for the nearest node to the input location and is reported as "depth to water table" and "water table elevation". If the selected node is saturated, then "water table elevation" gives the location of closest water table above the selected node and "depth to water table" is negative. Conversely, for an unsaturated node, "water table elevation" reports the location of closest water table below the selected node and "depth to water table" is positive. Output is written to the Tecplot ASCII file *prefix*`o.water_table.`**name**`.dat`. For each timestep in the flow solution, one line is written to the file. Each line contains the simulation time [T], depth to water table [L], and water table elevation [L].

●●●

## Report water table at node

1. **name** Descriptive name, up to 40 characters.
2. **node** Node number of the location at which to compute the water table.

Water table position is computed for the input node and is reported as "depth to water table" and "water table elevation". If the selected node is saturated, then "water table elevation" gives the location of closest water table above the selected node and "depth to water table" is negative. Conversely, for an unsaturated node, "water table elevation" reports the location of closest water table below the selected node and "depth to water table" is positive. Output is written to the Tecplot ASCII file *prefix*`o.water_table.`**name**`.dat`. For each timestep in the flow solution, one line is written to the file. Each line contains the simulation time [T], depth to water table [L], and water table elevation [L].

●●●

## Interpolate depth to gw table

This command causes **HydroGeoSphere** to use linear interpolation when computing the depth to groundwater table. It applies globally to the commands Report water table at xy, Report water table at xyz, and Report water table at node, as well as to the binary output files *prefix*`o.depth2gwt_pm.`*suffix*. By default, interpolation is not used.

●●●

#### 2.12.4.5 Fluid Exchange

## Report exchange for olf zones

This command generates a time series of fluid exchange flow rates [$L^3$ $T^{-1}$] for each zone in the overland flow domain. Output is written to the Tecplot ASCII file *prefix*-`o.fluid_exchange_olfz.dat`. If the simulation includes transport, then a time series of mass exchange flow rates [M $T^{-1}$] for each zone in the overland flow domain is written to the Tecplot ASCII output file *prefix*`o.mass_exchange_olfz_`*species*`.dat`, for each solute.

●●●

## Report exchange for chosen nodes

1. **label** Descriptive name, up to 80 characters.

This command generates a time series of fluid exchange flow rates [$L^3$ $T^{-1}$] for the currently active domain (overland flow or channel) and the current set of chosen nodes. Output is written to the Tecplot ASCII file *prefix*`o.fluid_exchange_`*domain*`.`**label**`.dat`, where *domain* is either *olf* or *chan*. Variables reported in the overland flow domain output file are:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Olf2PM` | [$L^3$ $T^{-1}$] | Rate that water moves into subsurface. |
| `PM2Olf` | [$L^3$ $T^{-1}$] | Rate that water moves out of subsurface. |
| `Net (Olf2PM - abs(PM2Olf))` | [$L^3$ $T^{-1}$] | Rate of change in subsurface storage. |

Variables reported in the channel domain output file are:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Chan2PM` | [$L^3$ $T^{-1}$] | Rate that water moves into subsurface. |
| `PM2Chan` | [$L^3$ $T^{-1}$] | Rate that water moves out of subsurface. |
| `Net (Chan2PM - abs(PM2Chan))` | [$L^3$ $T^{-1}$] | Rate of change in subsurface storage. |
| `Chan2Olf` | [$L^3$ $T^{-1}$] | Rate that water moves into surface. |
| `Olf2Chan` | [$L^3$ $T^{-1}$] | Rate that water moves out of surface. |
| `Net (Chan2Olf - abs(Olf2Chan))` | [$L^3$ $T^{-1}$] | Rate of change in surface storage. |

If the simulation includes transport, then a time series of mass exchange flow rates [M $T^{-1}$] for the overland flow domain is written to the Tecplot ASCII output file *prefix*`o.mass_exchange_olf.`**label**`.`*species*`.dat`, for each solute. Variables reported in this file are:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Olf2PM` | [M $T^{-1}$] | Rate that mass moves into subsurface. |
| `PM2Olf` | [M $T^{-1}$] | Rate that mass moves out of subsurface. |
| `Net (Olf2PM - abs(PM2Olf))` | [M $T^{-1}$] | Rate of change in subsurface mass. |

•••

#### 2.12.4.6 Fluid Volume

The following commands can be used to output information about the volume of water stored within a region of the surface and subsurface flow domains.

### Fluid volume for chosen elements

1. **name** Descriptive name for the volume, up to 40 characters.

This command computes the total volume of water stored within a set of chosen elements. The volume of stored water is reported for the porous media domain and optionally the surface flow domain if present. Output is written to the Tecplot ASCII file *prefix*`o.fluid_volume.`-**name**`.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Surface` | [$L^3$] | Volume of water stored in surface. |
| `Porous Media` | [$L^3$] | Volume of water stored in subsurface. |
| `Total` | [$L^3$] | Total volume of water stored in surface/subsurface. |

●●●

## Fluid volume for chosen elements by layer

1. **name** Descriptive name for the volume, up to 40 characters.

This command is similar to the command Fluid volume for chosen elements in that it computes the total volume of water stored within a set of chosen elements over all layers in the mesh. The volume of stored water is optionally reported for the surface flow domain if present. Output is written to the Tecplot ASCII file *prefix*`o.fluid_volume.`**`name`**`.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Surface` | [$L^3$] | Volume of water stored in surface. |
| `PM`$\ell$ | [$L^3$] | Volume of water stored in layer $\ell$ of subsurface. |
| `Total PM` | [$L^3$] | Total volume of water stored in subsurface. |
| `Total` | [$L^3$] | Total volume of water stored in surface/subsurface. |

Note that porous media layers are reported starting at the top layer ($\ell$ = number of layers) of the mesh and proceeding downwards to the bottom layer ($\ell = 1$).

●●●

## Fluid volume to tecplot

This command causes all element selections for the commands Fluid volume for chosen elements and Fluid volume for chosen elements by layer to be written to the Tecplot ASCII output file *prefix*`o.fluid_volume_selection.dat`.

●●●

## Compute water volume by zone

This command reports the water volume [$L^3$] stored within each porous medium zone and overland flow zone in the Tecplot ASCII files *prefix*`o.water_volume_pm_zone.dat` and *prefix*`o.water_volume_olf_zone.dat`, respectively.

●●●

## Compute water volume by area using shp file

1. **filename** Name of the shapefile without the file extension.

2. **id** Unique integer id ($\geq 1$) for the selected area.

Reports the water volume [$L^3$] stored within a set elements (porous medium domain) or faces (surface domain), depending on the currently active domain, whose centroids belong to the polygon defined by the shapefile (`.shp`). Polygon, PolygonM, and PolygonZ shape types are supported. Output is written to the Tecplot ASCII files *prefix*`o.water_volume_pm_area.dat` and *prefix*`o.water_volume_olf_area.dat`. Each time this command is issued, a new variable named "area**id**" is appended to the corresponding file.

●●●

#### 2.12.4.7 Fluid Flux Entering Volume

These commands can be used to compute the fluid flux between a set of contributing nodes and a set of active nodes. Output fluxes are reported for each individual domain (see Section 2.8.1) that is present in the model. Output is written to Tecplot ASCII files named *prefix*`o.fluid_entering_volume.`*name*`.dat`, where *name* is a unique user specified identifier for a volume, that report the following variables (depending on which domains are active):

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Porous Media` | [$L^3$ $T^{-1}$] | Porous medium fluid flux. |
| `Dual` | [$L^3$ $T^{-1}$] | Dual continuum fluid flux. |
| `Fractures` | [$L^3$ $T^{-1}$] | Discrete fracture fluid flux. |
| `Overland` | [$L^3$ $T^{-1}$] | Surface fluid flux. |
| `Channels` | [$L^3$ $T^{-1}$] | Channel fluid flux. |
| `Wells` | [$L^3$ $T^{-1}$] | Well fluid flux. |
| `Tiles` | [$L^3$ $T^{-1}$] | Tile fluid flux. |
| `Total` | [$L^3$ $T^{-1}$] | Total fluid flux among all active domains. |

Fluid fluxes are computed between active nodes (on the boundary of the volume) and contributing nodes (just outside the volume boundary) as illustrated in Figure 2.21. Fluxes are reported in terms of their positive and negative components (e.g., variables `Overland+` and `Overland-`) with the convention that a flux is positive if water is entering the volume and negative if water is leaving the volume.

## Flux volume output nodes

1. **vol_name** Unique identifier for the volume, up to 40 characters.

Chosen nodes are used to define a volume for which detailed fluid flux information is reported. Active nodes (the boundary of the volume) and contributing nodes (just outside

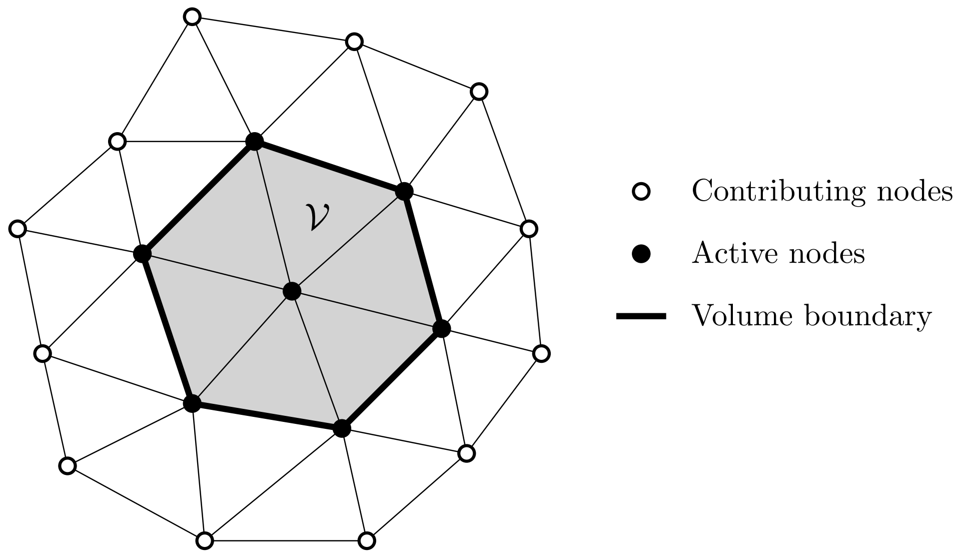


Figure 2.21: Active and contributing nodes for a volume $\mathcal{V}$.

the boundary of the volume) are defined automatically by searching for 3-D elements that contain a mixture of chosen and unchosen nodes. For such elements, chosen nodes are flagged as active and unchosen nodes are flagged as contributing.

●●●

In some cases, it may be necessary to restrict the computation of fluid flux to a slice of nodes. The following instructions can be used to do this:

## Slice flux output nodes from chosen

1. **vol_name** Unique identifier for the volume, up to 40 characters.

This instruction works in a similar way to Flux volume output nodes except that all currently chosen nodes are flagged as active nodes for the fluid flux calculation.

●●●

The preceding instruction must be used in conjunction with the following instruction:

## Slice flux contributing nodes from chosen

This instruction flags all currently chosen nodes as contributing nodes for the fluid flux calculation. Note that **grok** checks that both instructions are issued in the correct order and that there is at least one element that shares an active and contributing node.

●●●

## Vertical slice flux by layer

1. **vol_name** Unique identifier for the volume, up to 40 characters.
2. **contributing_name** Name of contributing node set, up to 40 characters.
3. **active_name** Name of active node set, up to 40 characters.
4. **minlayer, maxlayer** Minimum and maximum layers (inclusive), respectively.

This command is similar to the command combination Slice flux output nodes from chosen and Slice flux contributing nodes from chosen. For each node set (active or contributing) it generates a vertical slice over the input layer range (**minlayer** to **maxlayer**). The active and contributing node sets should be generated by the command Create node set. Each node set is treated as a 2-D node set by projecting all nodes to the bottom sheet of the model. The main difference with this command is that the contribution to the total fluid flux by each layer in the porous media domain is written to the output file. Porous media variables are reported with the layer number appended to their name. Any layers in the porous media domain that fall outside of the layer range are reported as `-9999999999` in the output file.

• • •

#### 2.12.4.8 Surface Flow Hydrographs

The following command can be used to define a hydrograph over a set of nodes in the surface domain.

## Set hydrograph nodes

1. **label** Descriptive name for the hydrograph, up to 80 characters.

Chosen nodes in the surface domain are flagged as hydrograph nodes, at which detailed total flow rate information is output at each timestep. Output is written to the Tecplot ASCII file *prefix*`o.hydrograph.label.dat` that reports the following variables:

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `Surface` | [$L^3$ $T^{-1}$] | Surface domain flow rate. |
| `Porous Media` | [$L^3$ $T^{-1}$] | Flow rate between surface and porous media domains. |
| `Total` | [$L^3$ $T^{-1}$] | Total of `Surface` and `Porous Media`. |
| `Channel` | [$L^3$ $T^{-1}$] | Channel domain flow rate. |
| `ChanLeak` | [$L^3$ $T^{-1}$] | Flow rate between channel and surface/porous media domains. |
| `ChannelTotal` | [$L^3$ $T^{-1}$] | Total of `Channel` and `ChanLeak`. |

Note that the variables `Channel`, `ChanLeak`, and `ChannelTotal` are present only if the model defines a channel flow domain.

●●●

#### 2.12.4.9 Polygon Tracking

The following commands can be used to output water balance information over a region in the surface flow domain contained within a polygon.

### Fluid mass balance for olf areas using shp file

1. **filename** Name of the shapefile without the file extension.
2. **id_name** Unique identifier, up to 40 characters.

This command causes **HydroGeoSphere** to output water balance information for the region of the surface flow domain that is contained within the polygon defined by the shapefile (`.shp`). Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.* The reported area consists of all elements in the surface flow domain whose centroid belongs to the polygon.

Output is written to the Tecplot ASCII file *prefix*`o.fluid_mass_balance.`**id_name**`.dat`. The first three columns of this file contain the simulation time [T], total precipitation rate [$L^3$ $T^{-1}$] on the polygon (variable `Precipitation`), and the rate of actual evapotranspiration [$L^3$ $T^{-1}$] from the polygon (variable `Total_AET`). The remaining columns may be grouped into blocks, with one block for the surface flow domain and one for each element layer in the porous media domain. Blocks are ordered starting with the surface flow domain and then proceeding downwards through each layer of the porous media domain. For example, if a model has ten node sheets, hence nine element layers, there would be ten blocks of columns in the output file. Column headings for the blocks would be indexed by 10 for the surface flow domain and from 9 for the uppermost porous media layer to 1 for the lowermost layer. The following table describes the variables reported within each block of columns.

| Variable | Units | Description |
|---|---|---|
| `ET` | [$L^3$ $T^{-1}$] | Rate of evapotranspiration from polygon. |
| `S` | [$L^3$ $T^{-1}$] | Rate of storage change within the polygon. |
| `QH+` | [$L^3$ $T^{-1}$] | Horizontal flux into the polygon. |
| `QH-` | [$L^3$ $T^{-1}$] | Horizontal flux out of the polygon. |
| `QV+` | [$L^3$ $T^{-1}$] | Vertical flux into an element layer of the polygon. |
| `QV-` | [$L^3$ $T^{-1}$] | Vertical flux out of an element layer of the polygon. |

●●●

## Fluid mass balance for olf areas using shp file by layer

1. **filename** Name of the shapefile without the file extension.
2. **id_name** Unique identifier, up to 40 characters.
3. **minlayer, maxlayer** Minimum and maximum layers (inclusive), respectively.

This command causes **HydroGeoSphere** to output water balance information for the region of the surface flow domain that is contained within the polygon defined by the shapefile (`.shp`). Polygon, PolygonM, and PolygonZ ($z$-coordinate ignored) shape types are supported. *The polygon may contain holes and consist of multiple parts but must not be self-intersecting.* The only difference between this command and Fluid mass balance for olf areas using shp file is that porous media domain output is restricted to the input layer range (**minlayer** to **maxlayer**). Note that any layers in the porous media domain that fall outside of the layer range are reported as `-9999999999` in the output file.

●●●

## Fluid mass balance for olf areas for chosen faces

1. **id_name** Unique identifier, up to 40 characters.

This command causes **HydroGeoSphere** to output water balance information for the region of the surface flow domain defined by the current set of chosen faces. See the command Fluid mass balance for olf areas using shp file for a detailed description of the output produced by this command.

●●●

## Fluid mass balance for olf areas for chosen faces by layer

1. **id_name** Unique identifier, up to 40 characters.
2. **minlayer, maxlayer** Minimum and maximum layers (inclusive), respectively.

This command causes **HydroGeoSphere** to output water balance information for the region of the the surface flow domain defined by the current set of chosen faces. The only difference between this command and Fluid mass balance for olf areas for chosen faces is that porous media domain output is restricted to the input layer range (**minlayer** to **maxlayer**). Note that any layers in the porous media domain that fall outside of the layer range are reported as `-9999999999` in the output file.

●●●

### 2.12.5 Transport

The following instructions affect the I/O for the transport solution.

#### Flux output nodes

1. **new_noutfc** Number of new output nodes desired.
2. **ioutfc(i), i=1,new_noutfc** Flux output node number.

Listed nodes are flagged to generate detailed mass flux [M $T^{-1}$] information to the file *prefix*`o.flm`. For each timestep in the transport solution, one line per flux output node per species will be written to the file which contains the species number, node number, time, concentration, mass flux and nodal coordinates. Such output can be imported into an editor (e.g., Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of concentration or flux versus time for a given species at a node.

For an example which uses flux output nodes, see Section 1.5.1 in the Verification Manual.

• • •

#### Flux output nodes from chosen

This instruction works in a similar way to Flux output nodes except that all currently chosen nodes are flagged as flux output nodes.

• • •

#### Plot concentration penetration depth

1. **threshold_conc** Threshold concentration [M $L^{-3}$].

This instruction causes **HydroGeoSphere** to write the distance from domain top to the elevation of the threshold concentration contour versus time to a file called *prefix*`o.penetration_depth`. This command is especially useful in conjunction with density-dependent flow simulations where the migration of a dense plume from the domain top into the aquifer versus time is of interest.

• • •

#### Plot maximum velocity

Causes **HydroGeoSphere** to write the maximum porous matrix velocity versus time to a file called *prefix*`o.maxvel_pm`. If fractures are present, the additional file *prefix*`o.maxvel_fr` is created, containing the maximum fracture velocity versus time data.

• • •

## Compute statistical properties of plume

This instruction causes **HydroGeoSphere** to write the statistical properties of solute plume in porous media at each time step (zeroth to fourth order moments). Output is written to the file *prefix*`o.Statistical_Plume_Props.dat`.

• • •

## Report nodal mass storage for chosen nodes

1. **label** Unique identifier, up to 40 characters.

Nodal mass storage values are accumulated over the set of chosen nodes. Output is written for each species to the Tecplot ASCII output file *prefix*`o.nodal_mass_storage.`**label**`.`*species*`.dat`. Each line of the file reports the simulation time [T], the nodal mass stored [M], mass decay (dissolved) rate [M $T^{-1}$], mass sorbed [M], and mass decay (sorbed) rate [M $T^{-1}$] for each domain that is present in the model (porous media, dual continua, fracture, surface flow; see Section 2.8.1).

• • •

## Fluid volume concentration threshold

1. **label** Unique identifier, up to 40 characters.

2. **threshold_conc(i), i=1,nspeciesmob** Concentration threshold [M $L^{-3}$] for each solute.

Reports the volume of fluid with concentration above and below a user specified concentration threshold for each solute and for all chosen nodes. Output is written to the Tecplot ASCII output file *prefix*`o.fluid_volume_conc.`**label**`.`*species*`.dat`, where *species* is the name of the solute. Each line of the file reports the simulation time [T], the volume of fluid [$L^3$] with concentration above and below the concentration threshold, and the total volume of fluid [$L^3$] for the porous media domain and surface flow domain if defined. Note that a no data value of $-9999$ is reported for the surface domain if it does not intersect with the chosen node set.

• • •

#### 2.12.5.1 Observation Wells and Points

Observation well/point commands (see Section 2.12.4.1) output timeseries information at a single node or a group of nodes lying on a line. Output is written to a Tecplot ASCII file named *prefix*`o.observation_well_conc.`*well_name*`.`*species*`.dat`, where *well_name* is a user specified identifier for an observation well/point and *species* is the species (i.e., solute) name. Depending on model setup, for example, the domains/processes defined, the following variables may be reported in the output file:

| Variable | Units | Domain | Description |
|---|---|---|---|
| `Time` | [T] | | Simulation time (transient) |
| `Cmob` | [M $L^{-3}$] | PM | Mobile zone concentration (dual porosity) |
| `Cimmob` | [M $L^{-3}$] | PM | Immobile zone concentration (dual porosity) |
| `C` | [M $L^{-3}$] | PM | Concentration |
| `Redox` | [M $L^{-3}$] | PM | Redox capacity |
| `Cd` | [M $L^{-3}$] | Dual | Concentration |
| `Co` | [M $L^{-3}$] | Overland | Concentration |
| `Cf` | [M $L^{-3}$] | Fracture | Concentration |
| `rho` | [-] | | Relative density (density dependent flow) |
| `mu` | [M $L^{-1}$ $T^{-1}$] | | Viscosity (density dependent flow) |

In addition, the $xyz$-coordinates, surface elevation at $(x, y)$, and node number are reported for each node that forms the observation well/point. Note that if a node does not belong to a given domain, then the no data value ($-999.0$) is reported.

#### 2.12.5.2 Solute Mass Balance

By default, solute mass balance information for each species is computed at each time step and written to the *prefix*`o.lst` file. Figure 2.22 illustrates some sample output as it appears in that file. The model output and the text in this section assume that the species is a solute and results are reported in terms of the mass of solute, with units of mass given by [M]. When the species is temperature, the energy balance is computed instead of the mass balance. In that case, when consulting the energy balance results, the user should replace mass [M] by energy, whose units are [M $L^2$ $T^{-2}$]. If the SI base units of kilogram, meter, second are specified for the simulation, the energy is expressed in Joule [J]. Furthermore, the species concentration [M $L^{-3}$] becomes temperature in degrees Celsius for temperature species.

```
*********************************************************************************
MASS BALANCE SPECIES  1 - Uranium
*********************************************************************************
 (Time =   1999.00999999978       Time-step =  9.999999776482582E-003 )
RATE OF MASS EXCHANGE                  IN                 OUT
```

```
TOTAL
   Fixed head nodes             100.0000006818      0.0000000000
   Fixed concentration nodes    150.0985515732      0.0000000000
   NET1 EXCHANGE RATE (IN-OUT)                                         250.0985522550

MASS ACCUMULATION                  COMPONENT         SUBTOTAL
   In storage:
      Porous medium               2.5001748233
      Porous medium (sorbed)  35749.9997988755
   ACCUMULATED                                   35752.4999736989
   Lost by decay:
      Porous medium               0.0000000708
      Porous medium (sorbed)      0.0010116896
   DECAYED                                           0.0010117604
   EXCHANGED (via chemical reactions)                0.0000000000
   INITIAL (stored at start of timestep)         35750.0000000000
   NET2 RATE OF ACCUMULATION                                           250.0985515100
      i.e. (ACCUMULATED -DECAYED +(-)EXCHANGED -INITIAL)/delta

   NET (since start of simulation)                 250.0985515100

MASS BALANCE ERROR
   Absolute: (NET1-NET2)                                                 0.0000007450
   Relative: (NET1-NET2)/(abs(NET1)+abs(NET2))/2.0                       0.0000000015
**********************************************************************************
```

Figure 2.22: Example of solute mass balance information written to file *prefix*`o.lst`.

Detailed mass balance information for each solute is also written to a Tecplot formatted ASCII output file named *prefix*`o.mass_balance_raw.`*species*`.dat` so that it can be easily visualized. For each timestep one line is written to the file. The number of columns in the file depends on the nature of the problem. For example, for problems with no overland flow component, no overland flow data will be written to the file. The first section of the raw mass balance output file gives the rate [M T$^{-1}$] at which mass is entering or leaving all model domains via various types of sources or sinks. The mass at the $i$th timestep, $M_i$, can be computed from the rate, $r_i$, via the equation

$$M_i = r_i(t_i - t_{i-1}), \quad i \geq 1,$$

where consecutive time values can be read from the table and by default $t_0 = 0$. The section begins by listing the net mass flux during a timestep for various boundary conditions and domains that represent an outlet for flow from the model. We note that for some boundary conditions the mass flux for each user specified boundary condition of that type is listed with the column heading consisting of its assigned name. This section of the file may include the following column headings:

- **Fixed head+, Fixed head-** Rate at which mass is entering or leaving the model domain via the head boundary condition [M T$^{-1}$].

- **Fluid flux+, Fluid flux-** Rate at which mass is entering or leaving the model domain via all specified flux or potential evapotranspiration boundary conditions [M $T^{-1}$].
- **Well+, Well-** Rate at which mass is entering or leaving the model domain via well nodes [M $T^{-1}$].
- **Tile+** Rate at which mass is entering the model domain via tile drain nodes [M $T^{-1}$].
- **Fixed conc+, Fixed conc-** Rate at which mass is entering or leaving the model domain via first-type transport boundary condition nodes [M $T^{-1}$].
- **Mass injected** Rate at which mass is entering the model domain via nodal mass flux boundary condition nodes [M $T^{-1}$].
- **Mass injection nodes** Rate at which mass is entering the model domain via nodal mass flux from concentration boundary condition nodes [M $T^{-1}$].
- **Specified Heat Flux** Rate at which energy is entering the model domain via specified heat flux boundary condition nodes [M $T^{-1}$].
- **Third type+, Third type-** Rate at which mass is entering or leaving the model domain via third-type transport boundary condition nodes [M $T^{-1}$].
- **CriticalDpth-** Rate at which mass is leaving the model domain via critical depth boundary condition nodes [M $T^{-1}$].
- **Tunnel-** Rate at which mass is leaving the model domain via tunnel boundary condition nodes [M $T^{-1}$].
- **ZeroDpthGrad-** Rate at which mass is leaving the model domain via the Zero Depth Gradient boundary condition nodes [M $T^{-1}$].
- **Seepage+, Seepage-** Rate at which mass is entering or leaving the model domain via seepage boundary condition nodes [M $T^{-1}$].
- **FreeDrain-** Rate at which mass is leaving the model domain via free drainage boundary condition nodes [M $T^{-1}$].
- **SimpleDrain+, SimpleDrain-** Rate at which mass is entering or leaving the model domain via simple drain boundary condition nodes [M $T^{-1}$].

The remaining part of this section may include the following column headings:

- **Decay of parent** Mass flux from parent species decay [M $T^{-1}$].
- **Zero source** Mass flux from zero-order sources [M $T^{-1}$].
- **First source+, First source-** Rate at which mass is entering or leaving the model domain from first-order sources [M $T^{-1}$].

- **Atm BC+, Atm BC-** Rate at which energy is entering or leaving all model domains from atmospheric boundaries [M $L^2$ $T^{-2}$]. The atmospheric boundaries only apply to the temperature species.
- **NET1 Source/Sink rate** Total rate of all mass entering or leaving all model domains through all sources and sinks [M $T^{-1}$].

The next section of the raw mass balance output file relates to the mass accumulation that occurred in the various model domains during a timestep. It may include the following column headings:

- **Porous media** Amount of mass stored in porous medium domain [M].
- **PM decayed** Amount of mass decayed (dissolved) in porous medium domain [M].
- **PM sorbed** Amount of mass sorbed in porous medium domain [M].
- **PM srb/dk** Amount of mass decayed (sorbed) in porous medium domain [M].
- **Immobile** Amount of mass stored in immobile zone (dual porosity) [M].
- **Isotope frac** Amount of mass stored in rock phase (isotope fractionation) [M].
- **Dual cont.** Amount of mass stored in dual continuum domain [M].
- **DC decayed** Amount of mass decayed (dissolved) in dual continuum domain [M].
- **DC sorbed** Amount of mass sorbed in dual continuum domain [M].
- **DC srb/dk** Amount of mass decayed (sorbed) in dual continuum domain [M].
- **Fractures** Amount of mass stored in fracture domain [M].
- **F decayed** Amount of mass decayed (dissolved) in fracture domain [M].
- **F sorbed** Amount of mass sorbed in fracture domain [M].
- **F srb/dk** Amount of mass decayed (sorbed) in fracture domain [M].
- **Overland** Amount of mass stored in overland flow domain [M].
- **O decayed** Amount of mass decayed (dissolved) in overland flow domain [M].
- **O sorbed** Amount of mass sorbed in overland flow domain [M].
- **O srb/dk** Amount of mass decayed (sorbed) in overland flow domain [M].
- **Tile** Amount of mass stored in tile drains [M].
- **Well** Amount of mass stored in well nodes [M].
- **ChemExchange** Rate of mass exchange resulting from chemical reactions [M $T^{-1}$].

- **Init mass** Amount of mass stored for all model domains at the start of a timestep [M].
- **Net since t0** Change in mass stored for all model domains since the start of the simulation [M].
- **NET2 Accum. rate** Rate of change of mass stored for all model domains [M T$^{-1}$].

If we let $M_i$ denote the total amount of mass stored (not accounting for any decayed mass) at time $t_i$ and let $M_{\text{decay},i}$ denote the total amount of decayed mass at time $t_i$, then after the $i$th timestep $[t_{i-1}, t_i]$ we have

$$\begin{aligned}\textbf{Init mass} &= M_{i-1}\\ \textbf{Net since t0} &= \sum_{k=1}^{i}(M_k + M_{\text{decay},k} - M_{k-1})\\ \textbf{NET2 Accum. rate} &= \frac{M_i + M_{\text{decay},i} - M_{i-1}}{t_i - t_{i-1}}\end{aligned}$$

The next section of the raw mass balance output file relates to the mass balance error. If $\pm Q$ is the total rate of change in mass from all sources and sinks (**NET1 Source/Sink rate**) and $\Delta S$ is the rate of change of mass storage (**NET2 Accum. rate**), then the mass balance error can be expressed in the following ways:

- **ERROR (NET1-NET2)** Absolute error [M T$^{-1}$]

$$\varepsilon = \pm Q - \Delta S$$

- **Error rel.** Relative error [-]

$$\varepsilon_R = \frac{2(\pm Q - \Delta S)}{|\pm Q| + |\Delta S|}$$

The final section of the raw mass balance output file gives the advective/dispersive solute exhchange between the porous media and dual continuum domain (if present). It consists of the following column headings:

- **Qm_DC2PM(Adv)** Advective solute exchange from dual continuum domain to porous medium domain [M T$^{-1}$].
- **Qm_DC2PM(Disp)** Dispersive solute exchange from dual continuum domain to porous medium domain [M T$^{-1}$].
- **Qm_PM2DC(Adv)** Advective solute exchange from porous medium domain to dual continuum domain [M T$^{-1}$].

- **Qm_PM2DC(Disp)** Dispersive solute exchange from porous medium domain to dual continuum domain [M $T^{-1}$].

Summary mass balance information for each solute is written to a Tecplot formatted ASCII output file named *prefix*`o.mass_balance_summary.`*species*`.dat` so that it can be easily visualized. This file consists of six columns labelled as follows:

1. **Time** Simulation time [T].
2. **Tmass** Total mass in the system [M].
3. **src:sink** Rate of change in mass entering and leaving all domains due to sources and sinks [M $T^{-1}$].
4. **dstore** Rate of change in mass stored in all domains [M $T^{-1}$].
5. **error** Error [M $T^{-1}$] (**src:sink** – **dstore**).
6. **error/Tmass*100.0** Normalized error [-].

In addition to being plotted with Tecplot, such output can be imported into an editor (e.g., Microsoft Excel) and sorted to facilitate, for example, the creation of a plot of mass balance error versus time for a given species.

Cumulative mass balance information is written to a Tecplot formatted ASCII output file named *prefix*`o.mass_balance_cumulative.`*species*`.dat` so that it can be easily visualized. This file consists of five columns labelled as follows:

1. **Time** Simulation time [T].
2. **In (cumulative)** Mass entering all domains for the current timestep [M].
3. **Out (cumulative)** Mass leaving all domains for the current timestep [M].
4. **Stored** Change in mass stored for the current timestep [M].
5. **In+Out-Stored** Error [M].

In addition to being plotted with Tecplot, such output can be imported into an editor (e.g., Microsoft Excel) and sorted to facilitate, for example, the creation of a plot of mass balance error versus time for a given species.

#### 2.12.5.3 Mass Flux Entering Volume

These commands (see Section 2.12.4.7) can be used to compute the mass flux between a set of contributing nodes and a set of active nodes. Output fluxes are reported for each individual domain (see Section 2.8.1) that is present in the model. Output is written to

Tecplot ASCII files named *prefix*`o.mass_entering_volume.`*name*`.`*species*`.dat`, where *name* is a unique user specified identifier for a volume and *species* is the species (i.e., solute) name, that report the following variables (depending on which domains are active):

| Variable | Units | Description |
|---|---|---|
| `Time` | [T] | Simulation time. |
| `C Matrix Avg` | [M $L^{-3}$] | Average porous medium concentration. |
| `Porous Media(A)` | [M $T^{-1}$] | Porous medium advective mass flux. |
| `Porous Media(D)` | [M $T^{-1}$] | Porous medium dispersive mass flux. |
| `C Macropore Avg` | [M $L^{-3}$] | Average dual continuum concentration. |
| `Macropore(A)` | [M $T^{-1}$] | Dual continuum dispersive mass flux. |
| `Macropore(D)` | [M $T^{-1}$] | Dual continuum dispersive mass flux. |
| `Discrete Fracture(A)` | [M $T^{-1}$] | Discrete fracture advective mass flux. |
| `Discrete Fracture(D)` | [M $T^{-1}$] | Discrete fracture dispersive mass flux. |
| `Overland(A)` | [M $T^{-1}$] | Surface advective mass flux. |
| `Overland(D)` | [M $T^{-1}$] | Surface dispersive mass flux. |
| `Instantaneous Total` | [M $T^{-1}$] | Total mass flux among all active domains. |
| `Cumulative Total` | [M] | Cumulative total mass among all active domains. |

Mass fluxes are computed between active nodes (on the boundary of the volume) and contributing nodes (just outside the volume boundary) as illustrated in Figure 2.21. Fluxes are reported in terms of their positive and negative components (e.g., variables `Overland(A)+` and `Overland(A)-`) with the convention that a flux is positive if mass is entering the volume and negative if mass is leaving the volume. For a volume $V$ and simulation time $t$, if $M_0(V)$ denotes the initial mass in $V$ and $M(t, V)$ denotes the cumulative total mass added to $V$ over the time interval $[0, t)$, then the total mass in $V$ at time $t$ is given by $M_0(V) + M(t, V)$.

### 2.12.6 Post Simulation Output

The commands in this section are executed following the completion of a simulation and cause **HydroGeoSphere** to report additional outputs that may be useful for model calibration. For example, the command Compute post simulation average instructs **HydroGeoSphere** to calculate the average value of a specified variable over a specified time interval.

## Compute post simulation average

1. **id_name** Unique identifier, up to 40 characters.

2. **filename** Name of the output file to read, up to 256 characters.

3. **var_name** Variable name, up to 40 characters. It must match the variable name in the variable line of the specified output file.

4. **tstart, tend** Start time [T] and end time [T] over which to compute the average.

This command computes the time-weighted average of the variable **var_name** in the output file **filename** over the time interval [**tstart**, **tend**]. Results are written to the ASCII output file *prefix*`o.avg_val_summary.dat`. This command applies to any **HydroGeoSphere** timeseries output file with the following generic format:

```
Title = "title"
Variables = "time", "var_name 1", "var_name 2", ..., "var_name n"
Zone T = "zone title"
t1 var11 var12 ... var1n
t2 var21 var22 ... var2n
 :   :     :         :
 :   :     :         :
```

For example, conforming output files include but are not limited to

- Observation point: *prefix*`o.observation_well_flow.`*well*`.dat`, *prefix*`o.observation_well_conc.`*well*`.`*species*`.dat`
- Fluid mass balance: *prefix*`o.water_balance.dat`
- Solute mass balance: *prefix*`o.mass_balance_raw.`*species*`.dat`, *prefix*`o.mass_balance_cumulative.`*species*`.dat`, *prefix*`o.mass_balance_summary.`*species*`.dat`
- Hydrograph: *prefix*`o.hydrograph.`*hydrograph*`.dat`

The command may be called more than once, with each call adding a new line to the generated output file.

●●●

# References

Allen, R. G., Pereira, L. S., Raes, D., and Smith, M. (1998). Crop evapotranspiration: guidelines for computing crop water requirements. *FAO Irrigation and Drainage Paper*, 56.

Canadell, J., Jackson, R. B., Ehrlinger, J. R., Mooney, H. A., Sala, O. E., and Schulze, E. D. (1996). Maximum rooting depth of vegetation types at the global scale. *Oecologia*, 108:583–595.

Cooley, R. L. (1983). Some new procedures for numerical simulation of variably-saturated flow problems. *Water Resour. Res.*, 19(5):1271–1285.

GDAL/OGR contributors (2024). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation.

Huyakorn, P. S., Wu, Y. S., and Park, N. S. (1994). An improved sharp-interface model for assessing NAPL contamination and remediation of groundwater systems. *J. Contam. Hydrol.*, 16:203–234.

Hwang, H.-T., Park, Y.-J., Sudicky, E. A., and Forsyth, P. A. (2014). A parallel computational framework to solve flow and transport in integrated surface-subsurface hydrologic systems. *Environ. Model. Softw.*, 61:39–58.

Kristensen, K. J. and Jensen, S. E. (1975). A model for estimating actual evapotranspiration from potential evapotranspiration. *Nordic Hydrol.*, 6(3):170–188.

Letniowski, F. W. and Forsyth, P. A. (1991). A control volume finite element method for three-dimensional NAPL groundwater contamination. *Int. J. Num. Meth. Fluids*, 13(8):955–970.

McLaren, R. G., Forsyth, P. A., Sudicky, E. A., VanderKwaak, J. E., Schwartz, F. W., and Kessler, J. H. (2000). Flow and transport in fractured tuff at Yucca Mountain: numerical experiments on fast preferential flow mechanisms. *J. Contam. Hydrol.*, 43:211–238.

Oswald, S. E. and Kinzelbach, W. (2004). Three-dimensional physical benchmark experiments to test variable-density flow models. *J. Hydrol.*, 290:22–42.

Partington, D., Brunner, P., Frei, S., Simmons, C. T., Werner, A. D., Therrien, R., and Fleckenstein, J. H. (2013). Interpreting streamflow generation mechanisms from integrated surface-subsurface flow models of a riparian wetland and catchment. *Water Resour. Res.*, 49:5501–5519.

Partington, D., Brunner, P., Simmons, C. T., Therrien, R., Werner, A. D., Dandy, G. C., and Maier, H. R. (2011). A Hydraulic Mixing-Cell method to quantify the groundwater component of streamflow within spatially distributed fully integrated surface water-groundwater flow models. *Environ. Model. Softw.*, 26:886–898.

Partington, D., Brunner, P., Simmons, C. T., Werner, A. D., Therrien, R., Maier, H. R., and Dandy, G. C. (2012). Evaluation of outputs from automated baseflow separation methods against simulated baseflow from a physically based, surface water-groundwater flow model. *J. Hydrol.*, 458–459:28–39.

Partington, D. J. (2013). *Fully integrated modelling of surface-subsurface flow processes: quantifying in-stream and overland flow generation mechanisms.* PhD thesis, The University of Adelaide, School of Civil, Environmental and Mining Engineering.

Pitzer, K. S. (1973). Thermodynamics of electrolytes. I. Theoretical basis and general equations. *J. Phys. Chem.*, 77:268–277.

Pitzer, K. S. (1991). Ion Interaction Approach: Theory and Data Correlation. In Pitzer, K. S., editor, *Activity Coefficients in Electrolyte Solutions*. CRC Press, 2nd edition.

Pitzer, K. S. and Mayorga, G. (1973). Thermodynamics of electrolytes. II. Activity and osmotic coefficients for strong electrolytes with one or both ions univalent. *J. Phys. Chem.*, 77:2300–2307.

Schilling, O. S., Gerber, C., Partington, D. J., Purtschert, R., Brennwald, M. S., Kipfer, R., Hunkeler, D., and Brunner, P. (2017). Advancing Physically-Based Flow Simulations of Alluvial Systems Through Atmospheric Noble Gases and the Novel $^{37}$Ar Tracer Method. *Water Resour. Res.*, 53(12):10465–10490.

Scurlock, J. M. O., Asner, G. P., and Gower, S. T. (2001). Worldwide Historical Estimates of Leaf Area Index. Technical Report ORNL/TM-2001/268, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

Stimpson, C. J., Ernst, C. D., Knupp, P., Pébay, P. P., and Thompson, D. (2007). The Verdict Geometric Quality Library. Technical Report SAND2007-1751, Sandia National Laboratories, Albuquerque, New Mexico.

Wang, J. S. Y. and Narasimhan, T. N. (1985). Hydrologic mechanisms governing fluid flow in a partially saturated, fractured, porous medium. *Water Resour. Res.*, 21(12):1861–1874.

# Appendix A

# Output Files

Unless otherwise stated, the files described in this section are ASCII formatted.

## A.1 Grok

These files are created during regular execution:

`array_sizes.default`
: See Section 2.1.4.

`grok.dbg`
: General purpose output file for debugging information.

These scratch files are created from scanning input files and removing comments, blank lines, and white-space:

`scratch_grok`
: Cleaned up copy of `.grok` file.

`scratch_dprops`
: Cleaned up copy of `.dprops` file.

`scratch_fprops`
: Cleaned up copy of `.fprops` file.

`scratch_mprops`
: Cleaned up copy of `.mprops` file.

`scratch_oprops`
: Cleaned up copy of `.oprops` file.

`scratch_etprops`
Cleaned up copy of `.etprops` file.

`scratch_cprops`
Cleaned up copy of `.cprops` file.

`scratch_WellProps`
Cleaned up copy of `.wprops` file.

`scratch_TileProps`
Cleaned up copy of `.tprops` file.

These files are created as data for a specific problem is processed. Most are binary files that are read later by **HydroGeoSphere**:

*prefix*`o.boundary_conditions`
Boundary condition data.

*prefix*`o.ci`
Initial nodal concentrations (binary).

*prefix*`o.coordinates_pm`
3-D subsurface mesh node coordinates (binary).

*prefix*`o.coordinates_dual`
3-D dual continua mesh node coordinates (binary).

*prefix*`o.coordinates_olf`
2-D surface flow mesh node coordinates (binary).

*prefix*`o.coordinates_frac`
2-D discrete fracture mesh node coordinates (binary).

*prefix*`o.coordinates_chan`
1-D channel flow mesh node coordinates (binary).

*prefix*`o.coordinates_well`
1-D well flow mesh node coordinates (binary).

*prefix*`o.coordinates_tile`
1-D tile flow mesh node coordinates (binary).

*prefix*`o.DecayRate_`*species*`.0001`
Elemental first-order decay constant for a solute (binary).

*prefix*`o.eco`
**grok** listing file.

*prefix*`o.ElemIbedFraction_pm.0001`
3-D subsurface mesh elemental fraction of compressible interbeds (binary).

*prefix*`o.ElemK_dual.0001`
3-D dual continua mesh elemental hydraulic conductivity (binary).

*prefix*`o.ElemK_pm.0001`
3-D subsurface mesh elemental hydraulic conductivity (binary).

*prefix*`o.ElemPor_pm.0001`
3-D dual continua mesh elemental porosity (binary).

*prefix*`o.ElemStor_pm.0001`
3-D subsurface mesh elemental specific storage (binary).

*prefix*`o.ElemTort_pm.0001`
3-D subsurface mesh elemental tortuosity (binary).

*prefix*`o.elemental_rill_storage.dat`
Elemental rill storage if Read rill storage from raster used.

*prefix*`o.elements_pm`
3-D subsurface mesh porous media element node lists (binary).

*prefix*`o.elements_dual`
3-D subsurface mesh dual continua element node lists (binary).

*prefix*`o.elements_olf`
2-D surface flow mesh element node lists (binary).

*prefix*`o.elements_frac`
2-D discrete fracture mesh element node lists (binary).

*prefix*`o.elements_chan`
1-D channel flow mesh element node lists (binary).

*prefix*`o.elements_well`
1-D well flow mesh element node lists (binary).

*prefix*`o.elements_tile`
1-D tile flow mesh element node lists (binary).

*prefix*`o.elements_et`
3-D subsurface mesh element ET zone, EDF, and RDF numbers (binary).

*prefix*`o.fac`
3-D subsurface mesh face node lists (binary).

*prefix*`o.fluid_volume_selection.dat`
Fluid volume element selections when Fluid volume to tecplot used.

*prefix*`o.gen`
General input data (binary).

*prefix*`o.hi`
Initial nodal heads (binary).

*prefix*`o.mesh_quality.dat`
Mesh quality metrics when Report mesh quality used.

*prefix*`o.olf.ElemArea_MB`
2-D surface flow mesh element selection when
Fluid mass balance for olf areas using shp file used (binary).

*prefix*`o.olf.WV_ElemArea`
2-D surface flow mesh element selection when
Compute water volume by area using shp file used (binary).

*prefix*`o.p_s_table.`*prop*`.dat`
Pressure-saturation table for a material property.

*prefix*`o.pm.WV_ElemArea`
3-D subsurface mesh element selection when
Compute water volume by area using shp file used (binary).

*prefix*`o.RegionalModelOutput`
Regional model node and element mappings (binary).

*prefix*`o.s_k_table.`*prop*`.dat`
Saturation-relative permeability table for a material property.

*prefix*`o.seg`
3-D subsurface mesh segment node lists (binary).

*prefix*`o.siz`
Array size data determined by **grok** and used by **HydroGeoSphere**

for run-time array allocation.

*prefix*`o.species`
Species (i.e. solute) data (binary).

*prefix*`o.time_varying_et_zones_raster.dat`
Time varying ET zone selection when
Time varying et zones from raster used.

*prefix*`o.time_varying_lai_raster.dat`
Time varying LAI selection when
Time varying lai from raster used.

*prefix*`o.time_varying_pm_zones.dat`
Time varying porous medium zone selection when
Time varying pm zones from ascii file used.

These files are created if the 2-D Random Fracture Generator (Section 2.3.4) is used:

*prefix*`o.rfrac.fractures`
Listing file with fracture zone information.

*prefix*`o.rfrac.apertures`
Listing file with fracture aperture information.

*prefix*`o.rfrac.lengths`
Listing file with fracture length information.

*prefix*`o.rfrac.orientations`
Listing file with fracture orientation information.

## A.2 HydroGeoSphere

These files are created during regular execution:

`hs.dbg`
General purpose output file for debugging information.

*prefix*`.output_variable.control`
Output variable control file.

`parallelindx.dat`
Parallel simulation details.

`restart_file_info.dat`
Restart information file.

`restart_state.dat`
Restart state data file (binary).

These files are created for use with the Run-Time Debug Utility described in Appendix B:

`debug.control`
You can edit this file to change the run-time debug behaviour.

`scratch_debug`
Cleaned up copy of `debug.control` file.

`progress.dat`
Tecplot formatted file of CPU time vs system time.

These files are created when running a simulation with Hydraulic Mixing Cells (see Section 2.10):

`hmc_debug.control`
HMC debug file for setting run-time parameters.

*prefix*`o.hmc_fractions.`*out_name*`.`*domain*`.dat`
HMC fraction timeseries for an output zone and domain.

`hmc\`*prefix*`o.hmc_restart`
HMC restart fraction data file (binary).

`hmc\`*prefix*`o.hmc_`*domain*`.`*frac_name*`.0001`
HMC named fraction over domain (binary).

`hmc\`*prefix*`o.hmc_`*domain*`.mixing_ratio.0001`
HMC mixing ratio over domain (binary).

`hmc\`*prefix*`o.hmc_`*domain*`.hmc_error.0001`
HMC fraction error balance over domain (binary).

These files are created for a specific problem as the simulation progresses:

*prefix*`o.avg_value_summary.dat`
Average value summary if Compute post simulation average used.

*prefix*`o.Bc.`*bcname*`.dat`
Tecplot formatted file for plotting a boundary condition.

*prefix*`o.cen`
Nodal concentrations at the end of a simulation which may be used as initial concentrations for a simulation restart (binary).

*prefix*`o.cvo`
Elemental velocity in channels.

*prefix*`o.dt`
Timestep sizes if Time step sizes to file used.

*prefix*`o.ET_Detailed_MB_`*bcname*`.dat`
Detailed ET information for each PET boundary condition if Output ET details used.

*prefix*`o.ET_Detailed_MB_debug_`*bcname*`.dat`
Detailed ET information for each PET boundary condition if run-time debug command Write ET detail information used.

*prefix*`o.flm`
Mass flux if Flux output nodes used.

*prefix*`o.flm_b`
Mass flux if Binary flux output nodes used (binary).

*prefix*`o.flm_sg`
Surface water/groundwater advective-dispersive mass flux.

*prefix*`o.flu`
Fluid flux if Flux output nodes used.

*prefix*`o.flu_b`
Fluid flux if Binary flux output nodes used (binary).

*prefix*`o.flu_sg`
Surface water/groundwater fluid flux.

*prefix*`o.fluid_entering_volume.`*volume*`.dat`
Mass flux information for a volume if slice fluxes are used.

*prefix*`o.fluid_exchange_chan.`*label*`.dat`
Fluid exchange flow rate for channel flow domain for chosen nodes.

*prefix*`o.fluid_exchange_olfz.dat`
Fluid exchange flow rate for each surface flow zone.

*prefix*`o.fluid_exchange_olf.`*label*`.dat`
Fluid exchange flow rate for surface flow domain for chosen nodes.

*prefix*`o.fluid_mass_balance.`*label*`.dat`
Surface flow mass balance for an area when
Fluid mass balance for olf areas using shp file used.

*prefix*`o.fluid_volume.`*label*`.dat`
Volume of stored water in a region of the surface/subsurface.

*prefix*`o.fluid_volume_conc.`*label*`.`*species*`.dat`
Volume of stored fluid in surface/subsurface domains with concentration above and below a concentration threshold for a solute.

*prefix*`o.heat_source`
Heat source at each time when Exponential zero order source is used.

*prefix*`o.hen`
Final heads for restart (binary).

*prefix*`o.hp.dat`
Heat pump injection/extraction temperature.

*prefix*`o.hydrograph.`*label*`.dat`
Total flux for a set of hydrograph nodes at each timestep.

*prefix*`o.isotherm.`*well*`.dat`
Observation well isotherm.

*prefix*`o.lst`
**HydroGeoSphere** run-time output listing file.

*prefix*`o.mass_balance_cumulative.`*species*`.dat`
Cumulative solute mass balance at each timestep.

*prefix*`o.mass_balance_raw.`*species*`.dat`
Mass balance information for a solute at each timestep.

*prefix*`o.mass_balance_summary.`*species*`.dat`
Solute mass balance summary at each timestep.

*prefix*`o.mass_entering_volume.`*volume*`.`*species*`.dat`
Mass flux information for a volume and solute if slice fluxes are used.

*prefix*`o.mass_exchange_olf.`*label*`.`*species*`.dat`
Mass exchange flow rate for a solute for each surface domain node selection defined by the command Report exchange for chosen nodes.
*prefix*`o.mass_exchange_olfz_`*species*`.dat`
Mass exchange flow rate for a solute for each surface flow zone.

*prefix*`o.maxvel_fr`
Maximum discrete fracture velocity.

*prefix*`o.maxvel_pm`
Maximum porous media velocity.

*prefix*`o.mf_interp`
Interpolated mass flux.

*prefix*`o.newton_info.dat`
Detailed Newton iteration information at each timestep for simulations with unsaturated flow conditions.

*prefix*`o.nodal_mass_storage.`*label*`.`*species*`.dat`
Mass storage at a node at each timestep.

*prefix*`o.obc`
Concentration for PEST calibration.

*prefix*`o.obs`
Head for PEST calibration.

*prefix*`o.observation_well_conc.`*well*`.`*species*`.dat`
Observation well concentration for a solute.

*prefix*`o.observation_well_conc_chem.`*well*`.`*chemical*`.dat`
Observation well concentration for chemical species.

*prefix*`o.observation_well_flow.`*well*`.dat`
Observation well/point flow solution output.

*prefix*`o.observation_well_silica_chemisty.`*well*`.dat`
Observation well concentration for silica solute.

*prefix*`o.observation_well_silica_chemisty_fracture.`*well*`.dat`
Observation well concentration for silica solute in fracture.

*prefix*`o.particle_trace.`*trace_index*`.dat`
Trace path for each particle when particle tracing is used.

*prefix*`o.particle_travel_time.csv`
Exit status and travel time for each particle when particle tracing is used.

*prefix*`o.penetration_depth.dat`
Distance from domain top to elevation of the threshold concentration.

*prefix*`o.Rain_SnowMelt_balance.dat`
Rain and snowmelt balance summary at each timestep when
Rain and snowmelt BC is used.

*prefix*`o.reservoir_`*bcname*`.dat`
Pressure head and storage at each timestep for a Reservoir boundary condition.

*prefix*`o.reservoir_spillway.`*bcname*`.dat`
Inflow, discharge, and storage change at each timestep for a
Reservoir with spillway boundary condition.

*prefix*`o.sim_time_report`
Simulation time report file if Generate time report is used.

*prefix*`o.simoutflux.dat`
Outlet outflux PDF for production zone.

*prefix*`o.simulation_progress.csv`
Simulation progress tracking when Track simulation progress is used.

*prefix*`o.snow_balance.dat`
Snow balance summary at each timestep when the Snowmelt BC is used.

*prefix*`o.soil_balance.dat`
Amount of water in the partially- and fully-saturated subsurface, and in the surface.

*prefix*`o.Statistical_Plume_Props.dat`
Statistical properties of plume.

*prefix*`o.tvo`
Elemental velocity in tiles.

*prefix*`o.water_balance.dat`
Fluid mass balance summary at each timestep.

*prefix*o.water_table.*label*.dat
Depth and elevation to water table at each timestep.

*prefix*o.water_volume_olf_area.dat
Water volume stored in each surface flow area when Compute water volume by area using shp file is used.

*prefix*o.water_volume_olf_zone.dat
Water volume stored in each surface flow zone when Compute water volume by zone is used.

*prefix*o.water_volume_pm_area.dat
Water volume stored in each porous media area. Compute water volume by area using shp file is used.

*prefix*o.water_volume_pm_zone.dat
Water volume stored in each porous media zone when Compute water volume by zone is used.

*prefix*o.wvo
Elemental velocity in wells.

*prefix*o.zone_fluid_balance.z.*label*.dat
Surface flow mass balance for a zone when Fluid mass balance for olf zones is used.

The following binary output files may be created for a specific problem at each output time, in this case number 0001, as the simulation progresses. For many of these files, their creation is controlled by the output variable control file, *prefix*.output_variable.control, which is generated automatically by **HydroGeoSphere** unless an existing file is found. These files are processed for visualization by **HSPLOT** (Appendix D) or **HGS2VTU** (Appendix E).

*prefix*o.absolute_ap_change.0001
Changes in fracture aperture.

*prefix*o.absolute_kxx_change.0001
Changes in nodal hydraulic conductivity in $x$-direction.

*prefix*o.absolute_kyy_change.0001
Changes in nodal hydraulic conductivity in $y$-direction.

*prefix*o.absolute_kzz_change.0001
Changes in nodal hydraulic conductivity in $z$-direction.

*prefix*o.absolute_por_change.0001
Changes in nodal porosity.

*prefix*`o.compact_pm.0001`
Porous media nodal compactions.

*prefix*`o.conc_chem.`*species*`.0001`
Chemical species concentration for a solute.

*prefix*`o.conc_dual.`*species*`.0001`
Dual continua concentration for a solute.

*prefix*`o.conc_frac.`*species*`.0001`
Discrete fracture concentration for a solute.

*prefix*`o.conc_olf.`*species*`.0001`
Surface flow concentration for a solute.

*prefix*`o.conc_pm.`*species*`.0001`
Porous media concentration for a solute.

*prefix*`o.conc_pm.`*species*`-Redox.0001`
Porous media redox capacity for a solute.

*prefix*`o.concsolid_pm.`*species*`.0001`
Porous media solid phase concentration for colloid.

*prefix*`o.concsolid_dual.`*species*`.0001`
Dual continua solid phase concentration for colloid.

*prefix*`o.DeltaZ_pm.0001`
Porous media nodal elevations.

*prefix*`o.depth_chan.0001`
Channel domain water depth.

*prefix*`o.depth_olf.0001`
Overland flow domain water depth.

*prefix*`o.depth2gwt_pm.0001`
Depth to groundwater table.

*prefix*`o.DiffPeclet_pm.0001`
Porous media element diffusion Peclet numbers if Output peclet number used.

*prefix*`o.DualExchFlux_olf.0001`
Exchange flux between surface flow and dual continua domains.

*prefix*`o.DualExchSolAdv_olf.`*species*`.0001`
Advective solute exchange flux between surface flow and dual continua domains.

*prefix*`o.DualExchSolDisp_olf.`*species*`.0001`
Dispersive solute exchange flux between surface flow and dual continua domains.

*prefix*`o.envhead_pm.0001`
Porous media environmental heads.

*prefix*`o.ETEvap_olf.0001`
ET surface evaporation.

*prefix*`o.ETPmEvap_olf.0001`
ET subsurface evaporation.

*prefix*`o.ETPmEvap3D_pm.0001`
ET subsurface evaporation for 3-D mesh.

*prefix*`o.ETPmTranspire_olf.0001`
ET subsurface transpiration.

*prefix*`o.ETPmTranspire3D_pm.0001`
ET subsurface transpiration for 3-D mesh.

*prefix*`o.ETActual_olf.0001`
Actual surface and subsurface ET.

*prefix*`o.ExchFlux_chan.0001`
Exchange flux between porous media, surface flow, and channel flow domains.
Positive fluxes are into the channel flow domain.

*prefix*`o.ExchFlux_dual.0001`
Exchange flux between porous media and dual flow domains.
Positive fluxes are into the porous media domain.

*prefix*`o.ExchFlux_frac.0001`
Exchange flux between porous media and discrete fracture domains.
Positive fluxes are into the discrete fracture domain.

*prefix*`o.Exchflux_olf.0001`
Exchange flux between porous media and surface flow domains.
Positive fluxes are into the surface flow domain.

*prefix*`o.ExchFlux_tile.0001`

Exchange flux between porous media and tile flow domains.
Positive fluxes are into the tile flow domain.

*prefix*`o.ExchFlux_well.0001`
Exchange flux between porous media and well flow domains.
Positive fluxes are into the well flow domain.

*prefix*`o.ExchFlux_pm2_chan.0001`
Exchange flux from porous media to channel flow domain.
Positive fluxes are into the channel flow domain.

*prefix*`o.ExchFlux_olf2_chan.0001`
Exchange flux from surface flow to channel flow domain.
Positive fluxes are into the channel flow domain.

*prefix*`o.ExchSolAdv_frac.`*species*`.0001`
Advective solute exchange flux between porous media and fracture domains.

*prefix*`o.ExchSolAdv_olf.`*species*`.0001`
Advective solute exchange flux between porous media and surface flow domains.

*prefix*`o.ExchSolDisp_frac.`*species*`.0001`
Dispersive solute exchange flux between porous media and fracture domains.

*prefix*`o.ExchSolDisp_olf.`*species*`.0001`
Dispersive solute exchange flux between porous media and surface flow domains.

*prefix*`o.ExchSol_dual.`*species*`.0001`
Solute exchange flux between porous media and dual continua domains.

*prefix*`o.flow_rate_olf.0001`
Rate of flow into surface domain nodes from neighboring nodes.

*prefix*`o.fluid_density.0001`
Fluid densities.

*prefix*`o.flux_olf.0001`
Specified flux over surface flow domain.

*prefix*`o.freeze_thaw_temp.0001`
Porous media temperature.

*prefix*`o.friction_olf.0001`
Surface flow friction.

*prefix*`o.head_chan.0001`
Channel flow head.

*prefix*`o.head_dual.0001`
Dual continua head.

*prefix*`o.head_frac.0001`
Discrete fracture head.

*prefix*`o.head_olf.0001`
Surface flow head.

*prefix*`o.head_pm.0001`
Porous media head.

*prefix*`o.head_tile.0001`
Tile flow head.

*prefix*`o.head_well.0001`
Well flow head.

*prefix*`o.ice_sat_dual.0001`
Dual continua ice saturation.

*prefix*`o.ice_sat_pm.0001`
Porous media ice saturation.

*prefix*`o.iconc_pm.`*species*`.0001`
Porous media immobile concentration for a solute.

*prefix*`o.incised_head_chan.0001`
Channel flow head corrected for nonzero incision depth.

*prefix*`o.krw_pm.0001`
Element-by-element relative permeability in porous media domain.

*prefix*`o.kxx.0001`
Nodal hydraulic conductivity in $x$-direction.

*prefix*`o.kyy.0001`
Nodal hydraulic conductivity in $y$-direction.

*prefix*`o.kzz.0001`
Nodal hydraulic conductivity in $z$-direction.

*prefix*`o.liquid_p.0001`
Liquid precipitation over surface flow domain (Rain and snowmelt BC).

*prefix*`o.lp_chan.0001`
Location probability for channel flow domain.

*prefix*`o.lp_frac.0001`
Location probability for discrete fracture domain.

*prefix*`o.lp_olf.0001`
Location probability for surface flow domain.

*prefix*`o.lp_pm.0001`
Location probability for porous media domain.

*prefix*`o.max_depth_chan.0001`
Channel domain maximum water depth.

*prefix*`o.max_depth_olf.0001`
Surface domain maximum water depth.

*prefix*`o.Peclet_pm.0001`
Porous media element Peclet numbers if Output peclet number used.

*prefix*`o.permafrost_frac.0001`
Discrete fracture permafrost.

*prefix*`o.permafrost_pm.0001`
Porous media permafrost.

*prefix*`o.pet_olf.0001`
Nodal potential ET prescribed by Potential evapotranspiration BC.

*prefix*`o.por_pm.0001`
Porous media porosity.

*prefix*`o.pressure_head_dual.0001`
Dual continua pressure head.

*prefix*`o.pressure_head_frac.0001`
Discrete fracture pressure head.

*prefix*`o.pressure_head_pm.0001`
Porous media pressure head.

*prefix*`o.pressure_head_tile.0001`
Tile flow pressure head.

*prefix*`o.pressure_head_well.0001`
Well flow pressure head.

*prefix*`o.q_dual.0001`
Dual-permeability Darcy flux.

*prefix*`o.q_pm.0001`
Darcy flux.

*prefix*`o.rain_olf.0001`
Liquid precipitation (including snowmelt) over surface flow domain.

*prefix*`o.rate_pm.0001`
Porous media reaction rates.

*prefix*`o.sat_dual.0001`
Dual continua saturation.

*prefix*`o.sat_frac.0001`
Discrete fracture saturation.

*prefix*`o.sat_pm.0001`
Porous media saturation.

*prefix*`o.snowcover.0001`
Elevation of the top of the snow cover over the surface flow domain
(Rain and snowmelt BC).

*prefix*`o.snowdepth.0001`
Snow depth over the surface flow domain (Rain and snowmelt BC).

*prefix*`o.snowmelt.0001`
Snowmelt as liquid equivalent over the surface flow domain (Rain and snowmelt BC).

*prefix*`o.soilfrost_pm.0001`
Porous media soil frost.

*prefix*`o.solid_p.0001`
Snowfall as liquid equivalent over the surface flow domain (Rain and snowmelt BC).

*prefix*`o.sw_pm.0001`
Element-by-element water saturation in porous media domain.

*prefix*`o.tv_zone_pm.0001`
Porous media time varying zonation.

*prefix*`o.tvk_pm.0001`
Porous media time varying hydraulic conductivity.

*prefix*`o.v_chan.0001`
Channel flow velocity.

*prefix*`o.v_dual.0001`
Dual continua velocity.

*prefix*`o.v_frac.0001`
Discrete fracture velocity.

*prefix*`o.v_olf.0001`
Surface flow velocity.

*prefix*`o.v_pm.0001`
Porous media velocity.

*prefix*`o.v_tile.0001`
Tile flow velocity.

*prefix*`o.v_well.0001`
Well flow velocity.

*prefix*`o.vwc_pm.0001`
Porous medium volumetric water content.

# Appendix B

# Run-Time Debug Utility

When using **HydroGeoSphere** to solve complex problems, especially non-linear ones, it is often the case that the end user would like to, for example, view intermediate results, modify input parameters or print out the contents of various matrices. Normally, the program developer would carry out such tasks with the aid of a debugger, which is normally supplied as part of a program development package. Unfortunately, most end users do not have access to such a package, and even if they did, running a code which has been compiled in debug mode is generally much slower than when it has been optimized for release to the public.

In order to address some of these problems, we have developed a run-time debug utility which operates on the optimized **HydroGeoSphere** executable and is able to be activated and deactivated interactively without halting the execution of the program.

At various points during program execution, **HydroGeoSphere** checks for the presence of a file called `debug.control` in the same directory as the *prefix*`.grok` file and, if found, performs certain actions as indicated in the file. If it is not found, a file will be created which contains instructions for performing run-time debug actions.

If you want to prevent **HydroGeoSphere** from performing run-time debugging (including checking the debug control file), you can do so using the instruction No runtime debug.

The head and tail of the `debug.control` file are shown in Table B.1.

```
debug off
! ------------------------------------- Pause execution
! pause timestep
! pause at time
!        10000.00000
! pause flow convergence loop
! ------------------------------------- Produce output
! write output files
! write saturated flow matrices

...etc...
```

```
! ------------------------------------ Adaptive timestep targets
! concentration change target
!            0.05000
! mass change target
!        0.10000E+21
! mass error target
!          100.00000
! minimum timestep multiplier
!            0.50000
! maximum timestep multiplier
!            2.00000
```

Figure B.1: Sample contents of `debug.control` file.

Lines beginning with the comment character (!) are ignored. When first generated, the only uncommented line is the first one, `debug off`, which causes **HydroGeoSphere** to run normally, and not to take any run-time debug actions.

The contents of the file depend on the nature of the problem which is being simulated. For example, if it has no transport component there will not be any transport-related information written to the file.

To activate the debug utility, just comment out the `debug off` instruction, and uncomment one or more of the remaining instructions as desired.

For example, if you modified and saved `debug.control` so it appeared as shown in Table B.2

```
!debug off
! ------------------------------------ Pause execution
 pause timestep
...etc...
```

Figure B.2: Modified `debug.control` file.

then **HydroGeoSphere** would pause at the end of the next timestep.

Note that if the `debug.control` file becomes corrupted, or if you modify the problem in some way (e.g., activate transport) you can automatically generate a fresh copy by deleting or renaming it before or during **HydroGeoSphere** execution.

The effect caused by uncommenting the various instructions in the `debug.control` file will now be described, and input data requirements will be discussed as required.

## Debug off

Ignore the rest of the contents of the `debug.control` file, whether or not they are uncommented. To activate the run-time debug feature, this line should be commented out.

• • •

## Pause timestep

Pause at the beginning of the next timestep and issue the following message on the screen:

```
DEBUG CONTROL: pause timestep, press a key to continue
```

This command is usually used to prevent other instructions, such as Write output files, from producing too much output.

• • •

## Pause at time

1. **time** Simulation time [T].

Proceed until the given simulation time is reached and then pause.

• • •

## Pause newton loop

Pause at the beginning of the next Newton iteration.

• • •

## Pause flow convergence loop

Pause at the beginning of the next flow solver iteration.

• • •

## Write output files

Treat each timestep as if it was defined in the *prefix*`.grok` file as an output time (i.e., using the instruction Output times). Head, concentration, saturation, etc. output files will be written at each timestep. This instruction is usually used in conjunction with Pause timestep.

• • •

## Watch node list

1. **node(i)...end** Node number list.

A list of node numbers for which you want detailed output.

●●●

For each watch node, the following instructions write detailed output to the `hs.dbg` file:

Write flow matrices
Write transport matrices
Write overland flow
Write fracture (dual node) flow

Detailed output usually consists of the coefficient matrix and right-hand side vector for the watch node.

When using a feature such as Write flow matrices for example, be aware that **HydroGeoSphere** may write a lot of information to disk, and so they should be activated and deactivated for only a few timesteps at a time.

## Write delval node info

Detailed information, including $xyz$-coordinates, head, saturation, and relative permeability, for the node corresponding to the absolute maximum value of the linear solve solution (delval) at each timestep is written to the *prefix*`o.lst` file.

●●●

## Write resval node info

Detailed information, including $xyz$-coordinates, head, saturation and relative permeability, for the node corresponding to the absolute maximum value of the Newton residual (resval) at each timestep is written to the *prefix*`o.lst` file.

●●●

## Write seepage face output to .lst file

Details of the seepage node calculations, including which nodes are currently acting as seepage nodes and the fluid flux exiting the domain at each active seepage node are written to the *prefix*`o.lst` file.

●●●

## Time progress

Causes **HydroGeoSphere** to write a Tecplot file which contains values of simulation time versus real time. The slope of this line is a measure of the efficiency of **HydroGeoSphere** and it can be used to gauge how effective changes in parameter values are in speeding up the simulation. The first time this instruction is executed, the file `progress.dat` is created, and data is written to it as the run progresses. The results for a given simulation are tagged with a date and timestamp. Results from subsequent simulations are appended to the file so they can be compared from run to run.

•••

## Write krw file

1. **filename** Name of the file to write the hydraulic conductivity values, at most 80 characters.

For each principal direction and for each element, outputs the product of the relative permeability and the saturated hydraulic conductivity to the file **filename**. This file can then be read in a subsequent simulation using the instruction Read elemental k from file.

•••

## Time format

1. **type** Integer value indicating the type of format to use when writing time values to output files. Acceptable values are:

| | |
|---|---|
| 1 | Fixed. |
| 2 | Scientific. |
| 3 | General. |

•••

## Mass balance format

1. **type** Integer value indicating the type of format to use when writing mass balance output to file. Acceptable values are:

| | |
|---|---|
| 1 | Fixed. |
| 2 | Scientific. |
| 3 | General. |

•••

## Nodal flow check

1. **flow_check** Logical value (T/F), which if true, turns on the nodal flow check feature. Otherwise, it turns it off.

Turns on/off the nodal flow check feature. By default this feature is turned off.

●●●

## Force timestep

1. **delta_t** Timestep [T].

Sets all subsequent timesteps to **delta_t**.

●●●

The following commands are identical to those used in the `.grok` input file and will just be listed here. They are used to modify parameter values while **HydroGeoSphere** is running. Note that if you change a parameter here, **HydroGeoSphere** will continue to use the new value even if you disable run-time debugging (i.e., uncomment instruction Debug off) or comment out the instruction that was used to change the value.

Flow solver convergence criteria
Flow solver relative convergence criteria
Flow solver detail
Flow maximum iterations
Transport solver convergence criteria
Transport solver relative convergence criteria
Transport solver detail
Transport solver maximum iterations
Newton maximum iterations
Newton absolute convergence criteria
Newton residual convergence criteria
Compute underrelaxation factor
Nodal flow check tolerance
Newton maximum update for head
Newton maximum update for depth
Newton absolute maximum residual
Newton maximum residual increase
Maximum timestep
Minimum timestep multiplier
Maximum timestep multiplier

The following commands can be used to set the values of the various targets used in the adaptive timestepping procedure.

Head change target

Saturation change target
Water depth change target
Newton target
Concentration change target
Mass change target
Mass error target

# Appendix C

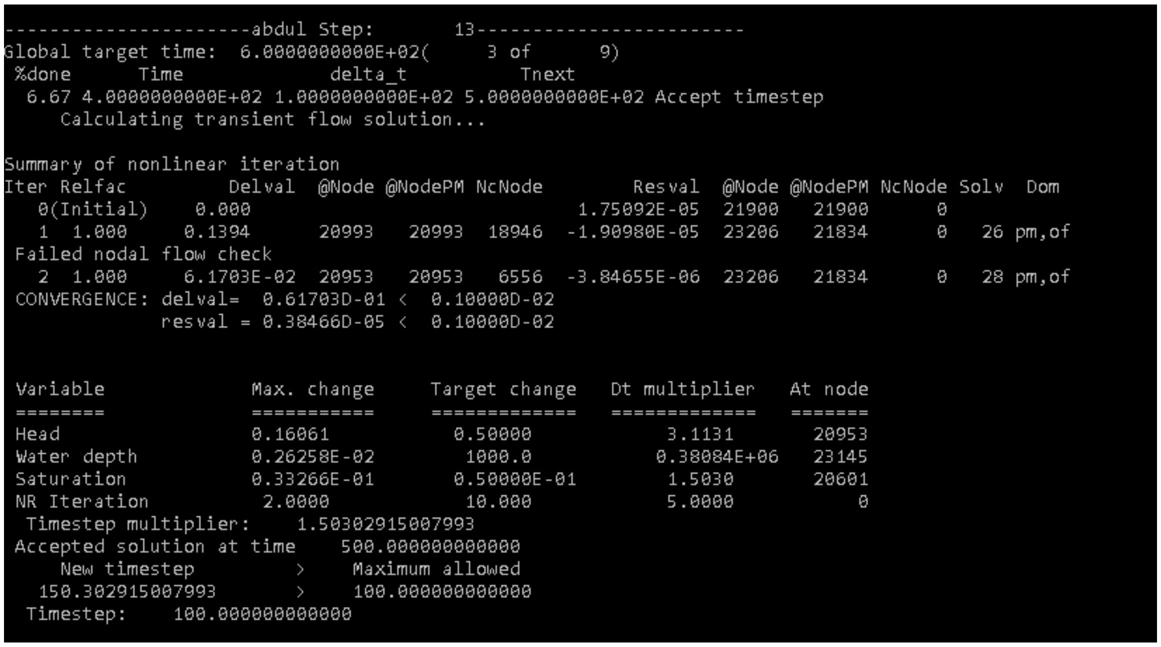
# Run-Time Timestep Output

In this section we discuss the run-time output produced by HGS at each timestep when computing the flow solution. This information is useful for tracking the progress and convergence behaviour of the simulation as well as for diagnosing and remedying any poor performance issues. Before we dig in, a brief overview of the solution process in HGS will aide our discussion. At each timestep, HGS employs the Newton–Raphson iterative scheme to solve the nonlinear equations that arise from discretizing the flow equations. If we denote the solution vector by $x$ (typically pressure head), then the nonlinear equations being solved can be expressed as $F(x) = 0$. In its most basic form, the Newton–Raphson iterative scheme for solving $F(x) = 0$ can be written as follows.

```
1: Set the initial guess for $x$
2: while not converged do
3:     Build the Jacobian matrix $J \leftarrow F'(x)$ and Newton residual $r \leftarrow F(x)$
4:     Solve the linear system $Js = -r$ for the Newton step vector $s$
5:     Update the relaxation factor $\lambda$ (if using), otherwise $\lambda \leftarrow 1$
6:     Update the solution: $x \leftarrow x + \lambda \cdot s$
7:     Check for convergence: $\|s\|_\infty \leq \tau_s$ or $\|r\|_\infty \leq \tau_r$
8: end while
```

We will use the symbols and steps defined here when discussing the run-time output.

Figure C.1 shows a screen capture of the run-time timestep output produced by HGS when running the Abdul verification problem. The output consists of three sections: simulation progress, summary of the nonlinear iteration, and the adaptive timestepping update. We will discuss each of these sections in detail.



```
----------------------abdul Step:       13------------------------
Global target time:  6.0000000000E+02(     3 of      9)
 %done      Time              delta_t          Tnext
  6.67 4.0000000000E+02 1.0000000000E+02 5.0000000000E+02 Accept timestep
     Calculating transient flow solution...

Summary of nonlinear iteration
Iter Relfac         Delval  @Node @NodePM NcNode        Resval  @Node @NodePM NcNode Solv  Dom
   0(Initial)    0.000                             1.75092E-05  21900   21900      0
   1  1.000     0.1394      20993   20993  18946  -1.90980E-05  23206   21834      0   26 pm,of
 Failed nodal flow check
   2  1.000     6.1703E-02  20953   20953   6556  -3.84655E-06  23206   21834      0   28 pm,of
 CONVERGENCE: delval=  0.61703D-01 <  0.10000D-02
              resval = 0.38466D-05 <  0.10000D-02


 Variable             Max. change     Target change   Dt multiplier   At node
 ========             ===========     =============   =============   =======
 Head                 0.16061          0.50000           3.1131       20953
 Water depth          0.26258E-02       1000.0          0.38084E+06   23145
 Saturation           0.33266E-01      0.50000E-01       1.5030       20601
 NR Iteration          2.0000           10.000           5.0000           0
  Timestep multiplier:    1.50302915007993
 Accepted solution at time    500.000000000000
     New timestep         >    Maximum allowed
   150.302915007993       >    100.000000000000
  Timestep:    100.000000000000
```

Figure C.1: Run-time timestep output for the Abdul verification problem.

## C.1 Simulation Progress

- **Global Target Time:** This is the next target time that HGS is working toward. In the brackets () we can see that it is the third such target time out of nine total.
- **%done:** The percentage of the simulation completed as measured from the initial start time to the previous successful simulation time. Note that the final target time is used as the end time of the simulation.
- **Time:** The previous successful simulation time.
- **delta_t:** The distance between the previous successful simulation time and the simulation time that is currently being computed.
- **Tnext:** The simulation time at which the solution is currently being computed.

## C.2 Summary of the Nonlinear Iteration

- **Iter:** The iteration number of the nonlinear solve step.
- **Relfac:** The underrelaxation factor $\lambda$ for the Newton iteration.
- **Delval:** The maximum absolute error of the current Newton iteration, given by $s_i$, where $i = \arg\max_j |s_j|$ and $s$ is the Newton step vector ($\Delta\psi_j^{r+1}$ in Equation (3.89)).

- **@Node:** The unknown number at which the absolute maximum occurs, i.e., $i = \arg\max_j |s_j|$.
- **@NodePM:** The mesh node number at which the absolute maximum value occurs.
- **NcNode:** The number of unknowns in $s$ that exceed the tolerance $\tau_s$ in absolute value (see the command Newton absolute convergence criteria).
- **Resval2:** The 2-norm of the Newton residual $\|r\|_2$. This value optionally appears in the nonlinear iteration summary, for example, when the command Newton residual 2-norm convergence criteria is used.
- **Resval:** The maximum absolute error of the current Newton iteration, given by $r_i$, where $i = \arg\max_j |r_j|$ and $r$ is the Newton residual vector ($f_i^r$ in Equation (3.89)).
- **@Node:** The unknown number at which the absolute maximum value occurs, i.e., $i = \arg\max_j |r_j|$.
- **@NodePM:** The mesh node number at which the absolute maximum value occurs.
- **NcNode:** The number of unknowns in $r$ that exceed the tolerance $\tau_r$ in absolute value (see the command Newton residual convergence criteria).
- **Solv:** The linear system $Js = -r$ is solved approximately via an iterative solver, by default, BiCGStab. This value is the number of iterations required by the linear solve to satisfy its convergence criteria (see the commands Flow solver convergence criteria, Flow solver relative convergence criteria, and Flow solver maximum iterations).
- **Dom:** The domains at which **Delval** and **Resval** are achieved, respectively.

The final lines of this section indicate that convergence was achieved after two iterations of the nonlinear solver. It shows the values of **Delval** and **Resval**, and their corresponding convergence tolerances. Note that only one such convergence criterion needs to be satisfied.

## C.3 Adaptive Timestep Update

- **Variable:** The quantity being tracked, i.e., head, water depth, saturation, etc.
- **Max. change:** The difference in the specified quantity between the last successful simulation time and the one being computed that corresponds to the absolute maximum change in that quantity.
- **Target change:** The targeted or expected change in the specified quantity over a single timestep as specified in **grok** or via the `debug.control` file (see Appendix B). For example, in the case of head, it would be the **grok** command Head control or the run-time debug command Head change target.
- **Dt multiplier:** The ratio of **Target change** to |**Max. change**| for the specified quantity.

- **At node:** The index at which the maximum absolute difference occurs.

At the end of the timestep output it indicates that the solution at time **Tnext** was accepted. It then shows how the next timestep is computed. The timestep multiplier, $\alpha_t$, is defined as the minimum of all the **Dt multiplier** values, restricted to the interval $[\alpha_{t,\mathrm{min}}, \alpha_{t,\mathrm{max}}]$ (see commands Minimum timestep multiplier and Maximum timestep multiplier). The next timestep, $\Delta t_{\mathrm{new}}$, is then computed from the previous timestep, $\Delta t$, according to the formula:

$$\Delta t_{\mathrm{new}} = \max\left(\min(\alpha_t \cdot \Delta t,\, \Delta t_{\mathrm{max}}),\, \Delta t_{\mathrm{min}}\right)$$

The minimum and maximum timesteps can be set by the commands Minimum timestep and Maximum timestep, respectively.

# Appendix D

# HSPLOT: Visualization Post-Processor

The utility program **HSPLOT** can be used to convert raw **HydroGeoSphere** output into files that are compatible with Tecplot. Similar to **grok** and **HydroGeoSphere**, when **HSPLOT** starts executing, it requires a problem prefix that can be entered interactively from the keyboard or supplied via the `batch.pfx` file. The first time you run **HSPLOT** for a specific problem, it creates the plot control file, *prefix*`.plot.control`, which contains a list of instructions that affect the output file format and contents. Note that if the plot control file becomes corrupted you can automatically generate a fresh copy by deleting or renaming it before executing **HSPLOT**. You can also generate a fresh copy, overwritting any existing copy, simply by running **HSPLOT** with the command-line flag `--make_plot_control`, for example,

```
hsplot --make_plot_control
```

We will now discuss the various sections of the plot control file and how they can be modified to produce the desired outputs.

## D.1 Initialization

The first section of the plot control file defines formatting options that apply globally to all output files generated by **HSPLOT**. This section is terminated by an `End` command. The first few lines of the plot control file are:

```
 tecplot mode
! tecplot binary mode
```

Lines beginning with the comment character (!) are ignored, and so by default, the file is set up to generate Tecplot ASCII formatted output (i.e., Tecplot mode instruction is uncommented). Uncommenting one of these instructions has the following effect:

---

## Tecplot mode

Causes **HSPLOT** to generate Tecplot ASCII formatted output. We recommend this output format for smaller models for which **HSPLOT** processing time is already fast and the generated output files are small. ASCII output files have the file suffix *.dat*.

●●●

After generating Tecplot ASCII output with **HSPLOT**, you can run the command-line utility `preplot` (available through your Tecplot installation) to convert the ASCII files to binary format. Doing so significantly reduces file size and improves Tecplot load times for larger models. For example, to generate a *.plt* file for the porous medium domain of the Abdul verification problem a user would issue

```
preplot abdulo.pm.dat
```

from the command line. The new binary file will be named `abdulo.pm.plt`. Note that if the ASCII output files are updated, then you will need to rerun the `preplot` utility. If you intend on converting **HSPLOT** ASCII output to binary format, we recommend that you instead consider writing directly to the SZL binary format via the following command.

---

## Tecplot binary mode

Causes **HSPLOT** to generate Tecplot binary formatted output (SZL binary format). We recommend this output format for moderate to large sized models in order to reduce **HSPLOT** processing time, the file size of the generated output files, and the Tecplot loading time when visualizing the files. Binary output files have the file suffix *.szplt*. See also the supporting commands Tecplot binary flush interval and Tecplot binary append.

●●●

The remaining commands in the initialization section are as follows.

---

## Tecplot binary flush interval

1. **interval** Integer distance between consecutive file flushes.

When writing Tecplot binary output, data is cached in memory and is only written to file when either the file is closed or flushed. Periodically flushing to file is important when working with larger models to avoid exhausting all available memory. By default, **HSPLOT** will flush to file after a Tecplot zone is defined, which corresponds to processing a single timestep worth of data. This command allows you to control when files are flushed by specifying the interval between consecutive file flushes. For example, a value of one corresponds to the default behavior, whereas a value of two causes **HSPLOT** to flush files after every two timesteps have been processed, and so on. Setting **interval** to a very large value will stop any file flushes from occurring. For the best possible performance, we recommend setting **interval** to as large a value as possible, taking into consideration the size of your model, the amount of output requested, and your machine's available memory. Note that flushing to file generates temporary files with the suffixes *.szhdr*, *.szdat*, *.szaux*, *.sztxt*, *.szgeo*, and *.szlab*, which are deleted automatically when the corresponding Tecplot file is closed. We recommend that any existing such files in your model directory be deleted prior to generating Tecplot binary output with **HSPLOT**.

●●●

## Tecplot binary append

When writing Tecplot binary output, **HSPLOT** can be run incrementally during a **Hydro-GeoSphere** simulation, allowing simulation results to be viewed on the fly. Each time **HSPLOT** is run in "append mode" it updates the temporary files with the suffixes *.szhdr*, *.szdat*, *.szaux*, *.sztxt*, *.szgeo*, and *.szlab*. These files may be combined into a single Tecplot *.szplt* file via the command-line utility `szcombine`, which should be available through your Tecplot installation. For example, to generate a *.szplt* file for the porous medium domain of the Abdul verification problem a user would issue

```
    szcombine abdulo.pm.szplt
```

from the command line. The new binary file will be named `abdulo.pm.szplt`. Note that it is the responsibility of the user to remove any existing Tecplot temporary files (as listed above) from a previous simulation prior to using this command. The user should also remove the binary file *prefix*`o.tec_append` if present.

●●●

## Truncate time domain

1. **t1, t2** Time range [T] of the domain.

Causes **HSPLOT** to restrict the time range of the output so that only output times that are between times **t1** and **t2** (inclusive) are included. Note that the first output time is always included.

●●●

## Number of digits for output file suffix

1. **suffix_len** Number of digits in the output file suffix, an integer between 4 and 10, inclusive.

Sets the number of digits in the output file suffix to **suffix_len**. By default the number of digits is set to four. Note that the number of digits set in **HSPLOT** should match the number set in **grok**, otherwise, **HSPLOT** will be unable to locate the output files.

● ● ●

After reading the initialization section, the plot control file is read until a terminating `End` command is encountered. Detailed output is written to the file *prefix*`o.hsplot.eco` each time **HSPLOT** is run.

## D.2 Domain Specific Output

The next sections of the plot control file control output for each of the domains: *pm, dual, frac, olf, well, chan, tile.* For example, the plot control file could include the following section for the porous media domain:

```
write pm domain file
! no pm heads
! no pm linear velocities (vx, vy, vz)
! no pm darcy velocities (vx, vy, vz)
! no pm elemental k
! no pm elemental porosity
! truncate pm domain
! -0.10000000E+21 0.10000000E+21
! -0.10000000E+21 0.10000000E+21
! -0.10000000E+21 0.10000000E+21
end
```

In general, each section begins with a general instruction that controls whether or not that domain's output is written, followed by a group of instructions that can be used to tailor the output for that domain, followed by a closing `End` command. For example, in the case of the porous media domain we have the following instruction.

### Write pm domain file...End

Causes **HSPLOT** to write the output files for the porous media domain. The default Tecplot formatted output file would contain all available data (i.e., heads, saturations, concentrations, velocities, etc.) at each output time and for the entire 3-D domain and would be named either *prefix*`o.pm.dat` in the case of ASCII output or *prefix*`o.pm.szplt` for binary output.

● ● ●

Similar commands exist for the other domains:

Write dual domain file...End
Write frac domain file...End
Write olf domain file...End
Write well domain file...End
Write chan domain file...End
Write tile domain file...End

In general, each command produces an output file named *prefix*`o.`*domain.ext*, where *domain* is the corresponding domain name and the file suffix *.ext* depends on the output format (ASCII or binary). Instructions that follow one of these commands prevent/allow **HSPLOT** from writing the named variables to the output file. For example, in the case of the porous media domain the following instructions are available:

No pm layer number
No pm column number
No pm element thickness
No pm heads
No pm pressure heads
No pm environment head
No pm saturations
No pm volumetric water content
No pm ice saturation
No pm freezing-thawing temperature
No pm linear velocities (vx, vy, vz)
No pm flux (qx, qy, qz)
No pm tvk (kx, ky, kz)
No pm concentrations
No pm immobile/isotope fractionation concentrations
No pm peclet number
No pm diffusion peclet number
No pm element k
No pm element porosity
No pm element storativity
No pm element tortuosity
No pm element fraction of compressible interbeds
No pm permafrost
No pm redox capacity
No pm et3d
No pm rain and snowmelt package
No pm depth to water table
No pm depth to water table 2d
No pm newton residual
No pm hmc fractions
Truncate pm domain
Truncate pm domain by layer

Some of these instructions are available for other domains by swapping “pm” in the command name for the appropriate domain name. In general, the presence of the instructions listed in the plot control file depends on the domain and on the nature of the simulation. For example, saturations would not be written unless the system is variably-saturated and the No pm saturations instruction would then not appear in the plot control file. Analogously, an instruction such as No frac depth to water table would not be written since it is not meaningful for the discrete fracture domain.

The commands Truncate pm domain and Truncate pm domain by layer require additional inputs as described below.

## Truncate pm domain

1. **x1, x2** *x*-range [L] of the domain.
2. **y1, y2** *y*-range [L] of the domain.
3. **z1, z2** *z*-range [L] of the domain.

Causes **HSPLOT** to restrict the output domain size to within the given *xyz*-ranges. This command is used to zero in on an interesting subregion of the 3-D domain or to reduce the size of the output file and speed up file I/O. For example,

```
truncate pm domain by layer
100.0     200.0
  0.0    1000.0
-0.10000000E+21 0.10000000E+21
```

would reduce the domain size to the *x*-range from 100 to 200, the *y*-range from 0 to 1000, but would leave the *z*-range at its full extent.

•••

## Truncate pm domain by layer

1. **x1, x2** *x*-range [L] of the domain.
2. **y1, y2** *y*-range [L] of the domain.
3. **lower, upper** Lower and upper layer numbers (inclusive).

Causes **HSPLOT** to restrict the output domain size to within the given *xy*-ranges and layer range. This command is used to zero in on an interesting subregion of the 3-D domain or to reduce the size of the output file and speed up file I/O. For example,

```
truncate pm domain by layer
100.0     200.0
  0.0    1000.0
  1         5
```

would reduce the domain size to the $x$-range from 100 to 200, the $y$-range from 0 to 1000, and would truncate the domain to layers 1–5.

● ● ●

The following are more specialized commands that can be used with the porous media domain.

## Compare heads 3d domain

1. **x1, x2** $x$-range [L] of the domain.
2. **y1, y2** $y$-range [L] of the domain.
3. **z1, z2** $z$-range [L] of the domain.

Causes **HSPLOT** to write Tecplot ASCII output files *prefix*`o.compare_heads.dat` and *prefix*`o.compare_heads_scatter.dat` that contain simulated versus observed head data. Only head data that falls within the restricted region defined by the $xyz$-ranges is included. Note that this instruction will only be present in the plot control file if an observed head data file called *prefix*`.observed_heads` is present in the directory that contains the *prefix*`.grok` file. The file *prefix*`.observed_heads` should be of the following form:

```
1255
8.29842E+05 2.41079E+06 2.85000E+02 2.85000E+02
8.34476E+05 2.39450E+06 2.86900E+02 2.95100E+02
...etc.
```

where the first line is the number of observed head points in the file, followed by one line for each point that contains the $xyz$-coordinates and observed head value at each point.

● ● ●

## Compare gw table elevation

Causes **HSPLOT** to write Tecplot ASCII output files *prefix*`o.compare_GWTable_Elevation.dat` and *prefix*`o.compare_GWTable_Elevation_scatter.dat` that contain simulated versus observed groundwater table elevation data. Note that this instruction will only be present in the plot control file if an observed groundwater table elevation file called *prefix*`.observed_heads` is present in the directory that contains the *prefix*`.grok` file.

●●●

## Compare depth to gw table

Causes **HSPLOT** to write Tecplot ASCII output files *prefix*`o.compare_Depth2GWTable.dat` and *prefix*`o.compare_D2GWT_scatter.dat` that contain simulated versus observed depth to groundwater data. Note that this instruction will only be present in the plot control file if an observed depth to groundwater file called *prefix*`.observed_d2gwt` is present in the directory that contains the *prefix*`.grok` file.

●●●

The instructions available for each of the other domains are listed in the following subsections.

### D.2.1 Dual Continua Output

No dual heads
No dual pressure heads
No dual saturations
No dual linear velocities (vx, vy, vz)
No dual concentrations
No dual immobile/isotope fractionation concentrations
No dual element k
No dual permafrost
No dual/pm exchange flux
No dual/pm solute exchange
Truncate dual domain
Truncate dual domain by layer

### D.2.2 Discrete Fracture Domain Output

No frac heads
No frac pressure heads
No frac saturations
No frac linear velocities (vx, vy, vz)
No frac concentrations
No frac immobile/isotope fractionation concentrations
No frac aperture
No frac/pm exchange flux
No frac/pm solute exchange
Truncate frac domain
Truncate frac domain by layer

### D.2.3 Surface Flow Domain Output

No olf heads
No olf max water depth
No olf linear velocities (vx, vy, vz)
No olf friction (nx, ny)
No olf concentrations
No olf immobile/isotope fractionation concentrations
No olf/pm exchange flux
No olf/pm solute exchange
No olf et
No olf local probability
No olf hmc fractions
No olf flow rates
Truncate olf domain
Truncate olf domain by layer

The following is a more specialized command that can be used with the surface flow domain.

#### Compare surface water depth

1. **x1, x2** $x$-range [L] of the domain.
2. **y1, y2** $y$-range [L] of the domain.
3. **z1, z2** $z$-range [L] of the domain.

Causes **HSPLOT** to write Tecplot ASCII output files *prefix*`o.compare_swdepth_olf.dat` and *prefix*`o.compare_swdepth_olf_scatter.dat` that contain simulated versus observed surface water depth data. Only surface water depth data that falls within the region defined by the $xyz$-ranges is included. Note that this instruction will only be present in the plot control file if an observed head data file called *prefix*`.observed_heads_olf` is present in the directory that contains the *prefix*`.grok` file.

●●●

### D.2.4 Tile Flow Domain Output

No tile heads
No tile pressure heads
No tile linear velocities (vx, vy, vz)
No tile concentrations
No tile immobile/isotope fractionation concentrations
No tile/pm exchange flux
No tile/pm solute exchange

Truncate tile domain
Truncate tile domain by layer

### D.2.5 Channel Flow Domain Output

No chan heads
No chan incised heads
No chan max water depth
No chan linear velocities (vx, vy, vz)
No chan concentrations
No chan immobile/isotope fractionation concentrations
No chan/pm exchange flux
No chan/pm solute exchange
No chan hmc fractions
Truncate chan domain
Truncate chan domain by layer

### D.2.6 Well Flow Domain Output

No well heads
No well pressure heads
No well linear velocities (vx, vy, vz)
No well concentrations
No well immobile/isotope fractionation concentrations
No well/pm exchange flux
No well/pm solute exchange
Truncate well domain
Truncate well domain by layer

## D.3 General Output Control

Additional commands are available that allow you to further tailor the output files produced by **HSPLOT**. Each of these commands should be issued at most once within the plot control file.

### Simulation time unit conversion factor

1. **time_offset, time_unit_factor** Time offset [$T_o$] and scaling factor [$T_o$ $T^{-1}$].

Causes **HSPLOT** to update the time stamp $t$ embedded within each output file it processes via the equation

$$t_{\text{new}} = \textbf{time\_offset} + \textbf{time\_unit\_factor} \cdot t$$

Note that the time unit $T_o$ may or may not be the same as T.

●●●

## Plot pm zone isopach structure

Causes **HSPLOT** to generate the Tecplot formatted output file *prefix*`o.pm_Isopach`*.ext* that contains porous media domain isopach data. The file suffix *.ext* is *.dat* for ASCII output and *.szplt* for binary output. Note that this command requires a surface flow domain to be present.

●●●

## Plot pm depth below ground surface

Causes **HSPLOT** to generate the Tecplot formatted output file *prefix*`o.pm_DepthToSurf`*.ext* that contains the depth of the porous media domain below ground surface. The file suffix *.ext* is *.dat* for ASCII output and *.szplt* for binary output. Note that this command requires a surface flow domain to be present.

●●●

## Interpolate depth to gw table

Causes **HSPLOT** to use linear interpolation when computing depth to groundwater. By default, interpolation is not used.

●●●

# Appendix E

# HGS2VTU: Visualization Post-Processor

The utility program **HGS2VTU** can be used to convert **HydroGeoSphere** binary output files and mesh files into various formats:

- Unstructured UGRID VTU files (readable by ParaView).
- NetCDF UGRID files.
- Tecplot binary (SZL) files.
- Tecplot ASCII files.
- CSV point cloud files.

It can also optionally be used to convert **HydroGeoSphere** Tecplot ASCII time series output files to CSV format (see command line option `--time-series-csv`).

At a bare minimum **HGS2VTU** requires the following HGS model files:

- Porous medium mesh files: *prefix*`o.elements_pm` and *prefix*`o.coordinates_pm`
- **grok** `.gen` file: *prefix*`o.gen`

Without these files no output can be generated. For each additional domain defined by your model (*dual, olf, frac, well, tile, chan*) the corresponding mesh files are required. In addition to the mesh files, **HGS2VTU** reads the binary output files generated by your model, e.g., *prefix*`o.head_pm.0001`, *prefix*`o.head_pm.0002`,...etc. and incorporates their data as variables in the output files it generates.

Running **HGS2VTU** is easy. Open a console and type:

```
hgs2vtu
```

to generate unstructured UGRID VTU output files (the default) for each domain defined by your model. The exit status of **HGS2VTU** is 0 if successful and nonzero otherwise. The behavior of **HGS2VTU** can be customized by a number of command-line options that are discussed below.

## E.1 Command Line Options

- `--prefix` Sets the name of the model prefix. If not provided **HGS2VTU** will attempt to read the model prefix from a `batch.pfx` in the working directory. Should that fail, the user will be prompted to provide the model prefix as console input.

- `--rev-num` Sets the HGS revision number of the HGS version that was used to generate the binary output files. The revision number is required by **HGS2VTU** to determine variable units and solute types among other things. If not provided, **HGS2VTU** will attempt to read the HGS revision number from the *prefix*`o.eco`, *prefix*`o.lst`, or *prefix*`o.hsplot.eco` files. Should that fail, **HGS2VTU** will terminate with an error.

- `--csv` Sets the output format to be a CSV point cloud. This format generates separate output files for nodal and cell-centered data, with one set of files per output time.

- `--netcdf` Sets the output format to be NetCDF UGRID. This format generates one file per output time following the UGRID Conventions. A current limitation of the NetCDF UGRID format is that it does not support the fracture domain. In order to generate NetCDF UGRID output, a user must supply additional information via the `--metafile` option discussed below. If this option is not specified, then a default metadata file named `meta_default.yml` will be generated followed by termination of **HGS2VTU**.

- `--metafile <filename>` Sets the filename of the metadata file that is required by the NetCDF UGRID format. The metadata file is a YAML file with the following simple format:

```
---
# Attributes are used to set user defined metadata that is added to
# the NetCDF files as attributes. Attribute names should start with a
# letter and be composed of letters, numbers, and underscores. You can
# issue this command multiple times to add different attributes.
# (optional)
Attribute: [<name>,<value>]

# Specifies the CRS code, e.g., EPSG:4326, which is a required to set
# the projection for your spatial coordinates. For backwards The CRS
# code can also be specified via "EPSG code: <value>". Note that since
# the coordinate transformation is done by proj, any CRS format that
# is accepted by proj should be suitable.
# (mandatory)
CRS code: <value>

# UTC date string, e.g., "1970-01-01T00:00:00Z", that defines the
# starting date of your simulation. It must be given in terms of local
# time. The start date is used together with the epoch date to define
# a time offset in hours from the epoch date. The time offset is added
# to the model output time to define a timestamp in hours since the
# epoch date for each NetCDF file.
# (mandatory)
Start date: <value>

# UTC date string, e.g., "1970-01-01T00:00:00Z", against which the
# timestamp of each output file is defined. It must be given in terms
# of local time. We recommend using an epoch date of 1970-01-01 to
# avoid potential issues caused by date to time point conversions.
# (optional, default is 1970-01-01T00:00:00Z)
Epoch date: <value>

# Defines the "z_reference" attribute in the NetCDF files which should
# always be "meters above sea level" (masl).
# (mandatory)
Z reference: <value>

# If `true` or `t`, then the porous medium domain and dual continuum
# domain NetCDF files are formatted as full 3D outputs according to the
# UGRID conventions. Otherwise, if `false` or `f`, then the porous medium
# and possibly the dual continuum domain (if layered) NetCDF files are
# formatted as 3D layers according to the UGRID conventions.
# (optional, default is false, i.e., layered 3D output)
Full 3D output: <value>
```

- `--tecplot`, `--tecplot-ascii` Sets the output format to be Tecplot ASCII. This format generates a single file that contains all output times.
- `--tecplot-szplt` Sets the output format to be Tecplot binary (SZL). This format generates a single file that contains all output times.
- `--tec-flush-int <intvl>` Sets the interval as the number of output times between consecutive file flushes for the Tecplot binary format. When writing Tecplot binary output, data is cached in memory and is only written to file when either the file is closed or flushed. Therefore, periodically flushing to file is important when working with larger models to avoid exhausting all available memory. By default the flush interval is set to one, which corresponds to flushing the file after each output time.

Setting a larger flush interval should result in better performance at the cost of higher memory usage.

- `--vtu` Sets the output format to be unstructured UGRID VTU (readable by ParaView), which is the default. This format generates one file per output time.

- `--time-series-csv` Causes **HGS2VTU** to convert HGS Tecplot ASCII time series output files to a corresponding CSV file, e.g., *prefix*`o.water_balance.dat` $\rightarrow$ *prefix*`o.water_balance.csv`. Conversion is done after processing the binary output files. Note that processing the binary output files can be skipped by either commenting or deleting all lines of the *prefix*.`plot.control` file. (Use the `--make-plot-control` option if you need to generate a fresh plot control file.)

- `--time-series-netcdf` Causes **HGS2VTU** to convert HGS Tecplot ASCII time series output files to a single NetCDF file named *prefix*`o.time_series.nc` that follows the CF-1.6 Conventions. Conversion is done after processing the binary output files. Currently, only a subset of HGS time series output files are supported by this option. Note that processing the binary output files can be skipped by either commenting or deleting all lines of the *prefix*.`plot.control` file. (Use the `--make-plot-control` option if you need to generate a fresh plot control file.)

- `--multi-file` Causes **HGS2VTU** to generate multiple files when converting HGS Tecplot ASCII time series output files to NetCDF format. For example, all hydrograph output files would be converted to the file *prefix*`o.hydrograph.nc`. By default, all HGS time series data is written to the single output file *prefix*`o.time_series.nc` when converting to NetCDF.

- `--unlimited-time` Causes **HGS2VTU** to specify an unlimited time dimension when converting HGS Tecplot ASCII time series output files to NetCDF format. By default, a fixed time dimension is used. Note that this option will likely result in faster conversion of the time series files to NetCDF at the cost of higher memory usage.

- `--nc-compress <chunk> <deflev>` Causes **HGS2VTU** to use chunking along the time dimension with the specified chunk size and compression with the specified deflate level when converting HGS Tecplot ASCII time series output files to NetCDF format. The chunk size is a non-negative integer and the deflate level is an integer between 0 and 9, with 2 being a good choice. Note that specifying a value of zero for either input disables that input. By default, no chunking or compression is applied (equivalent to `--nc-compress 0 0`).

- `--append <time index>` Causes **HGS2VTU** to append additional output starting at the specified time index $(1, 2, 3, \dots)$. Suppose you run a partial simulation that generates output data with time indexes: 0001, 0002, 0003 and convert the data with **HGS2VTU**. Then later your model generates additional data: 0004, 0005, 0006, . . . You can append this data to your existing output files with **HGS2VTU** by using the option `--append 4`. One caveat is that when using the Tecplot binary format you need to specify the `--append` option without any arguments the first time you run **HGS2VTU**. Otherwise, the Tecplot binary output files will be closed on termination of **HGS2VTU** making them impossible to append to at a later date.

- `--interpolate-d2gw` Causes **HGS2VTU** to use linear interpolation when computing depth to groundwater. Otherwise, by default the depth to groundwater is defined as the depth at which nodal pressure head changes sign.

- `--gwrecharge` Causes **HGS2VTU** to compute groundwater recharge from the overland flow domain and optionally the channel domain if defined resulting in the variable "gw_recharge" is added to the overland flow domain output files. This option requires the following HGS binary output files:

  - *prefix*`o.ExchFlux_olf.XXXX`
  - *prefix*`o.ETPmTranspire_olf.XXXX`
  - *prefix*`o.ETPmEvap_olf.XXXX`
  - *prefix*`o.ExchFlux_pm2_chan.XXXX` (if channel domain is defined)

  Groundwater recharge is computed according to the following formula:

  $$\text{gw\_recharge} = \max(-\text{Exch\_olf}, 0) + \max(-\text{Exch\_chan}, 0) + \text{Evap\_olf} + \text{Trans\_olf}$$

  where by convention Evap_olf and Trans_olf are negative, and negative overland flow and channel exchange fluxes means infiltration to the subsurface.

- `--clip-shape <filename>` Causes **HGS2VTU** to keep only those elements whose 2D centroids $(x_c, y_c)$ belong to the polygon defined by a shapefile (`.shp`) with the given filename. The polygon defined by the shapefile must be in the same projection as the data unless the option `--shape-proj` is also specified.

- `--shape-proj <source> <target>` Causes **HGS2VTU** to transform the coordinates of the shapefile defined by option `--clip-shape` from the source CRS to the target CRS. The source and target CRS strings are, for example, EPSG codes or WKT strings. Since **HGS2VTU** uses PROJ to do the transformation, a complete discussion of source and target CRS string formats can be found here.

  When using shapefiles to clip the model domain, **HGS2VTU** expects polygon coordinates to be given in units of meters (without reprojection or after reprojection via `--shape-proj`) regardless of the data length units. Consequently, the units of all 2D element centroids $(x_c, y_c)$ will be converted to meters when determining if they belong to a polygon.

- `--clip-layer <lower> <upper>` Causes **HGS2VTU** to clip the porous medium mesh layers to the layer range [lower, upper].

- `--make-plot-control` Causes **HGS2VTU** to generate a fresh plot control file named *prefix*.`plot.control` and exit. Note that if **HGS2VTU** cannot find a plot control file during its regular execution, then it will generate one automatically. The plot control file read by **HGS2VTU** can be used interchangeably with the plot control file read by **HSPLOT**. It contains a list of model domains and their associated variable names, and can be used to restrict which domains/variables are added to the output files. For example, consider the following snippet from the plot control file of a model that defines the overland flow domain:

```
end
write pm domain file
! no pm depth to water table
! no pm heads
end
write olf domain file
! no olf depth to water table
! no olf heads
end
```

By default both domains and all variables will be written to the output files generated by **HGS2VTU**. However, if you uncomment the line `!  no pm heads`, then porous medium head data would not be added to the porous medium output files. Alternatively, if you comment out the line `write pm domain file` and its associated end-statement, `end`, then the entire porous medium domain would be skipped.

The plot control file can also be used to control which time series output files are processed by **HGS2VTU** via the following block of commands:

```
write time series file
! no ts average value summary
! no ts bc flux
      :
      :
! no ts water volume pm zone
end
```

- `--sim-time-range <start> <end>` Causes **HGS2VTU** to truncate the simulation output times that are processed to the time range [start, end], which is in the same time units as the data.
- `--sim-time-shift <duration>` Causes **HGS2VTU** to subtract the time duration from the simulation output times when computing the timestamp for an output file. The time duration must be in the same time units as the data. For example, if the time duration is 1000 s and an HGS binary output file was written at an output time of 3600 s, then the data from that file will correspond to the time 2600 s in the **HGS2VTU** output. Note that any truncation of the simulation time range, for example, by the option `--sim-time-range` is based on the unshifted output times and is not affected by this option.

- `--utc-time-range <start> <end>` Causes **HGS2VTU** to truncate the simulation output times that are processed to the UTC date range [start, end]. UTC dates are expected to have the format `YYYY-MM-DDThh:dd:ssZ` and must be expressed in terms of local time. To use this option you must supply the model start date in a metadata file via the option `--metafile` described above.
- `--unit-system <usys>` Sets the unit system, where usys may be any one of the following options: `kg-m-s`, `kg-m-min`, `kg-m-h`, `kg-m-d`, `kg-m-y`, `kg-cm-s`, `kg-cm-min`, `kg-cm-h`, `kg-cm-d`, `kg-cm-y`. The unit system is used to define the units for all variables written by **HGS2VTU**. In newer versions of HGS this information is read from the *prefix*`o.gen` file, making this option unnecessary. Note that the unit system specified by this command will override any unit system read from the *prefix*`o.gen` file.
- `--verbose` Enables verbose logging, which causes **HGS2VTU** to generate a log entry for each binary output file that is converted. By default, **HGS2VTU** generates a single log entry for each output time that is processed.
- `--help` Causes **HGS2VTU** to display the help text and exit.
- `--version` Causes **HGS2VTU** to display the version information and exit.

## E.2 Output

Diagnostic information including any warning or error messages are written to the console as well as to a log file named *prefix*`o.hgs2vtu.eco`. Output filenames generated by **HGS2VTU** depend on the selected output format:

- Unstructured UGRID VTU files: *prefix*`o.`*domain.time_index*`.vtu`
- NetCDF UGRID files: *prefix*`o.`*domain.time_index*`.nc`
- Tecplot binary (SZL) files: *prefix*`o.`*domain*`.szplt`
- Tecplot ASCII files: *prefix*`o.`*domain*`.dat`
- CSV point cloud files: *prefix*`o.`*domain*`.nodal.`*time_index*`.csv` (nodal quantities) and *prefix*`o.`*domain*`.cell_centered.`*time_index*`.csv` (cell centered quantities)
- CSV time series: *prefix*`o.`*file_root*`.csv`
- NetCDF time series: *prefix*`o.time_series.nc` for single file output, *prefix*`o.`*file_root*-`.nc` for multifile output

where the *time_index* corresponds to the binary output file *suffix*, .e.g., 0001, 0002, . . . .

# Appendix F

# GMS File Formats

The following description is taken from the GMS Reference Manual, Version 1.1. GMS divides files into logical units called cards. The first component of each card is a short name which serves as an identifier. The rest of the line contains information associated with the card. Some cards can use multiple lines.

## F.1 Two-Dimensional Meshes (Slices)

The instructions Read gms 2d grid and 2D mesh to gms read and write 2-D mesh data in GMS format, respectively. The portion of the 2-D mesh file format recognized by **grok** is as follows:

```
MESH2D                          ! File type identifier
E3T id n1 n2 n3 mat             ! 3-node triangle
E4Q id n1 n2 n3 n4 mat          ! 4-node quadrilateral
ND id x y z                     ! Nodal coordinates
```

The cards E6T (6-node triangles) and E8Q (8-node quadrilaterals) are not recognized by **grok**. The file must consist of a single element type; mixed element types will result in an error.

The card types used in the 2-D mesh file are as follows.

| | |
|---|---|
| **Card Type** | MESH2D |
| **Description** | File type identifier. Must be on first line of file. No Fields. |
| **Required** | YES |

| | |
|---|---|
| **Card Type** | E3T |
| **Description** | Defines a 3-node (linear) triangular element. |
| **Required** | NO |
| **Format** | E3T id n1 n2 n3 mat |
| **Sample** | E3T 283 13 32 27 4 |

| **Field** | **Variable** | **Value** | **Description** |
|---|---|---|---|
| 1 | id | + | The id of the element. |
| 2–4 | n1–n3 | + | The nodal incidences of the elements ordered counterclockwise. |
| 5 | mat | + | The material id of the element. |

| | |
|---|---|
| **Card Type** | E4Q |
| **Description** | Defines a 4-node (linear) quadrilateral element. |
| **Required** | NO |
| **Format** | E4Q id n1 n2 n3 n4 mat |
| **Sample** | E4Q 283 13 32 27 30 4 |

| **Field** | **Variable** | **Value** | **Description** |
|---|---|---|---|
| 1 | id | + | The id of the element. |
| 2–5 | n1–n4 | + | The nodal incidences of the elements ordered counterclockwise. |
| 6 | mat | + | The material id of the element. |

| | |
|---|---|
| **Card Type** | ND |
| **Description** | Defines the coordinates of a node. |
| **Required** | NO |
| **Format** | ND id x y z |
| **Sample** | ND 84 120.4 380.3 5632.0 |

| **Field** | **Variable** | **Value** | **Description** |
|---|---|---|---|
| 1 | id | + | The id of the node. |
| 2–4 | x, y, z | ± | The nodal coordinates. |

## F.2 ASCII Scalar Data Set Files

The instruction Zone by layer can read a nodal data set in order to define a variable surface (usually an elevation for the *z*-coordinate) for the base of the 3-D grid or the top of a layer. This file should be written in GMS ASCII format.

The ASCII file format recognized by **grok** is as follows:

```
DATASET                         ! File type identifier
...etc...
ND n                            ! Number of data values
...etc...
TS time                         ! Time step of the following data
val_1                           ! Scalar data values
val_2

...etc...

val_n
```

In this case, the value `n` should correspond to the number of nodes. Currently, the first line of the file must contain the string `DATASET`, followed at some point by a `ND` card and then a `TS` card. Other other cards may be present in the file (e.g., `STAT`) but they will be ignored. You should not include status flag information in files to be read by **grok**. **grok** only reads one set of scalar data values per file.

The card type formats are as follows.

| | |
|---|---|
| **Card Type** | DATASET |
| **Description** | File type identifier. Must be on first line of file. No Fields. |
| **Required** | YES |

| | |
|---|---|
| **Card Type** | ND |
| **Description** | Defines the number of data values per time step. This number should correspond to the total number of nodes in the 2-D slice being used to generate the 3-D mesh. |
| **Required** | YES |
| **Format** | ND n |
| **Sample** | ND 4 |

| **Field** | **Variable** | **Value** | **Description** |
|---|---|---|---|
| 1 | n | + | The number of nodes per 2-D slice. |

| | |
|---|---|
| **Card Type** | TS |
| **Description** | Defines a set of scalar values associated with a timestep. |
| **Required** | YES |
| **Format** | TS time<br>$\text{val}_1$<br>$\text{val}_2$<br>$\vdots$<br>$\text{val}_n$ |
| **Sample** | TS 0.0<br>34.5<br>74.3<br>48.3<br>72.9 |

| **Field** | **Variable** | **Value** | **Description** |
|---|---|---|---|
| 1 | time | ± | The time step value. Not used by **grok**. |
| 2–(n+1) | val | ± | The scalar values for each item. |

# Appendix G

# Raster File Formats

Raster data consists of a set of values defined on the grid cells of a rectangular grid with uniform spacing in the $x$- and $y$-directions. The raster coordinates need not coincide with model mesh nodes and can even fall outside the model domain. **HydroGeoSphere** uses the open source software GDAL (GDAL/OGR contributors, 2024) to read raster data. GDAL supports a wide range of raster formats including ASCII, GeoTiff, and netCDF to name a few. On input of a raster file to **grok**, its format is determined automatically by GDAL, with an error being issued for an unsupported format. When working with rasters in **HydroGeoSphere**, it is important to keep in mind that raster data must use the same projection as the model data and no attempt will be made to convert raster data from a different projection.

**HydroGeoSphere** attempts to store raster data using the same data type as the raster, if possible, and uses a 64-bit floating point data type otherwise. Since Fortran does not support unsigned integers, all unsigned integer data is cast to signed integer data of the same size, e.g., `uint32_t` $\rightarrow$ `int32_t`. In doing so, there exists the potential for loss of information caused by integer overflow when reading rasters with unsigned integer data. Therefore, users are encouraged to use signed integer data types for all integer data. For rasters with unsigned integer data, it is the user's responsibility to ensure that their data (including the no data value) fits within the representable range of the corresponding signed data type. It is also important to note that since the **HydroGeoSphere** raster reading interface is defined in terms of a 64-bit floating point data type, all raster data (including the no data value) must fit within the representable range of that data type, which for integers is the range $[-2^{53}, 2^{53}]$.

A useful feature of some raster formats (e.g., GeoTiff) is that they support multiple layers of data called *bands*. Thus, a single raster file could, for example, contain forcing data at distinct times by assigning the data at each time to a distinct raster band. A number of **grok** commands have been designed to leverage this functionality, including:

- Elevation from raster
- Initial head from raster

- Map property from raster for chosen elements
- Read zones from raster
- Map zones from raster
- Read zones from raster, dominant class
- Map zones from raster, dominant class
- Choose faces top from raster
- Time raster table

These commands allow the user to specify the band number (an integer value $\geq 1$) as an optional input argument that appears after the raster filename. If a filename contains spaces, then it must be enclosed by double quotes (“”) to ensure correct parsing of the filename and band number by **grok**. For example:

```
elevation from raster
".\my path with spaces\surface elevation.asc" 2
```

These commands may also be used for raster formats that do not support bands (e.g., ASCII) in which case the band number will simply be ignored. In cases where a command accepts raster input but no band number argument is provided, a default value of one will be assumed.

Another nice feature of using GDAL is that raster data is automatically clipped to the bounding box of the mesh surface. Thus, rasters whose extents are much larger than the mesh itself use only as much memory as is necessary for your model domain.

# Index