



PLPROC

An example

A GMDSI tutorial

by John Doherty and Rui Hugman



PUBLISHED BY

The National Centre for Groundwater Research and Training
C/O Flinders University
GPO Box 2100
Adelaide SA 5001
+61 8 8201 2193

DISCLAIMER

The National Centre for Groundwater Research and Training, Flinders University advises that the information in this publication comprises general statements based on scientific research. The reader is advised and needs to be aware that such information may be incomplete or unable to be used in any specific situation. No reliance or actions must therefore be made on that information without seeking prior expert professional, scientific and technical advice.

PREFACE

The Groundwater Modelling Decision Support Initiative (GMDSI) is an industry-funded and industry-aligned project focused on improving the role that groundwater modelling plays in supporting environmental management and decision-making.

Over the life of the project, GMDSI will produce a suite of tutorials. These are intended to assist modellers in setting up and using model-partner software in ways that support the decision-support imperatives of data assimilation and uncertainty quantification. Not only will they focus on software usage details. They will also suggest ways in which the ideas behind the software which they demonstrate can be put into practice in everyday, real-world modelling practice.

GMDSI tutorials are designed to be modular and independent of each other. Each tutorial addresses its own specific modelling topic. Hence there is no need to work through them in a pre-ordained sequence. That being said, they also complement each other. Many employ variations of the same synthetic case and are based on the same simulator (MODFLOW 6). Utility software from the PEST suite is used extensively to assist in model parameterization, objective function definition and general PEST/PEST++ setup. Some tutorials focus on the use of PEST and PEST++, while others focus on ancillary issues such as introducing transient recharge to a groundwater model and visualization of a model's grid, parameterization, and calculated states.

The authors of GMDSI tutorials do not claim that the workflows and methodologies that are described in these tutorials comprise the best approach to decision-support modelling. Their desire is to introduce modellers, and those who are interested in modelling, to concepts and tools that can improve the role that simulation plays in decision-support. Meanwhile, the workflows attempt to demonstrate the innovative and practical use of widely available, public domain and commonly used software in ways that do not require extensive modelling experience nor an extensive modelling skillset. However, users who are adept at programming can readily extend the workflows for more creative deployment in their own modelling contexts.

We thank and acknowledge our collaborators, and GMDSI project funders, for making these tutorials possible.

Rui Hugman

John Doherty

CONTENTS

1. Introduction	5
2. Some Basics	6
2.1 MODFLOW-USG Grid Specification File	6
2.2 Node Data Table File	6
2.3 Using PARAVIEW for Display	7
2.4 A Closer Look at Model Zonation	9
3. PLPROC	11
3.1 Reading Model Grid Specifications	11
3.2 Pilot Points	12
3.3 Interpolation from Pilot Points to Model Grid Nodes	13
3.4 Evaluating Vertical Hydraulic Conductivities	14
3.5 Writing Model Input Files	15
4. Displaying in PARAVIEW	16
4.1 Hydraulic Property Data	16
4.2 Adding Pilot Points to the Picture	17

1. INTRODUCTION

“PLPROC” stands for “Parameter List PROCessor”. PLPROC is designed to facilitate calibration of large numerical models by serving as a model-independent pre-processor for such models. Through a combination of:

- bulk manipulation of parameters contained in user-defined parameter lists;
- assignment of values to members of one list based on values of non-congruent lists;
- an ability to read parameter values from files that employ a number of different formats; and
- an ability to write parameter values to model input files of arbitrary construction using template files that include embedded functions;

PLPROC allows a modeller to create and manipulate parameters that inform hydraulic properties that are represented in a numerical model. In doing this, PLPROC supports PEST in facilitating model-based decision-support which enshrines the principle that a model should encapsulate what we know while quantifying what we do not.

The current document and accompanying files offer a gentle introduction to PLPROC. They demonstrate the use of PLPROC in parameterization of a MODFLOW-USG model with multiple zones and layers. This demonstration does not comprise a comprehensive review of all functionality available in PLPROC. Nor does it illustrate the most recent additions to PLPROC functionality. Nevertheless, it should illuminate the basics of PLPROC usage. At the same time, it provides a reader with some insights into how to configure PLPROC for use in his/her own modelling context.

The current tutorial departs from the style of other GMDSI tutorials. All necessary files are provided ready-made. Explanations of files, and descriptions of workflows, are provided in the current document. By following instructions provided herein, a reader can run several utilities to regenerate these files and visualise model outputs. The present tutorial employs a MODFLOW-USG model. Other GMDSI tutorials are based on variations of a MODFLOW 6 model. Some of these tutorials also address the use of PLPROC. PLPROC workflows are similar, regardless of model type.

As well as PLPROC, this tutorial demonstrates several other utilities from the PEST Groundwater Utilities Suite. These include USG2VTK, USGADDZCOORD, USGGRIDLAY and USGPROP2TAB1. Executable versions of all programs belonging to the PEST [Groundwater Utilities](#) Suite can be downloaded from the PEST web site. However, to make this tutorial easy, executable versions of programs that are needed for its completion are provided in the tutorial folder itself. See documentation of the PEST Groundwater Utilities Suite for full descriptions of their use. Before commencing the tutorial, make sure that executable versions of these programs are copied to your machine (these are the “*.exe” files). Ideally, they should be stored in a folder that is cited in your computer’s PATH environment variable. Alternatively, they can simply be copied to your working folder.

To make full use of this tutorial the reader should have [PARAVIEW](#) (or a similar 3D visualisation package) installed. PARAVIEW is open source and freely available through the world wide web. The present tutorial demonstrates the use of utilities that enable visualisation of properties and system states associated with a MODFLOW-USG model. Similar utilities that work with MODFLOW 6 models are demonstrated in other GMDSI tutorials.

2. SOME BASICS

2.1 MODFLOW-USG Grid Specification File

File *example.gsf* is a “grid specification file” for MODFLOW-USG. This is not an official USGS file; however, it is written by graphical user interfaces such as Groundwater Vistas which support both MODFLOW-USG and PEST. Standard MODFLOW-USG input files feature no real-world coordinates. Furthermore, it is generally not possible to even know the shape of a MODFLOW-USG grid by reading the contents of a MODFLOW-USG DISU file.

A MODFLOW-USG grid specification file provides the coordinates not only of grid nodes (in the second part of the file), but also of grid cell vertices (in the first part of the file). Vertex coordinates are required for plotting of the grid, and for other tasks such as particle tracking and interpolation from node locations (where heads are calculated) to the locations of observation wells. If you inspect file *example.gsf* using a text editor, you will find that the first number on the second line of this file is “91644”. This is the number of nodes in the model grid.

example.disu is a MODFLOW-USG discretization file for the same model. It is not very interesting, and we will not be using it. However, you can inspect this file if it interests you. If you do, you will find the number “91644” again as the first entry on its first line.

2.2 Node Data Table File

Inspect file *example.bas*. This is the BASIC package input file for our MODFLOW-USG model. The model has 4 layers. IBOUND values are provided for these 4 layers. Our first task will be to extract IBOUND values for all model cells and record these values in a “node data table file”. Files of this type are employed by a number of members of the PEST Groundwater Utilities Suite. A node data table file lists nodes in its first column; one or more values are ascribed to all of these nodes in subsequent columns. We will now build a node data table file in which an IBOUND value (which in this case is either 1 or 0) is assigned to each node.

Run program USGPROP2TAB1 from the PEST Groundwater Utilities Suite, responding to its prompts as follows.

```
Enter name of MODFLOW-USG model input file: example.bas
```

```
Enter text string for array recognition in this file: ibound
```

```
Are these arrays integer or real? [i/r]: i
```

```
Enter total number of nodes in grid/network: 91644
```

```
Enter name for node data table file: ibound.ndt
```

```
Enter value to assign to uncited nodes: 0
```

Inspect file *ibound.ndt*. It is readily apparent that model node numbers are listed in the first column of this file, while node IBOUND values are listed in its second column.

Now inspect file *zones.ndt*. This is also a node data table file. However, in this case the integers in the second column are either 0 or a number between 3 and 10; a value of 0, once again, indicates an inactive cell. Positive values are zone values. This file could be written in the manner described above if IBOUND values are also zone numbers; thus, you could specify zonation for a model grid using your normal MODFLOW-USG graphical user interface. Alternatively, you could assign zone numbers to other arrays (integer or real) linked to the MODFLOW-USG model within that interface, and then have the interface write these arrays to a standard MODFLOW-USG input file. They could then be

extracted from that file using the USGPROP2TAB1 utility in the manner described above. Alternatively, the USGPROP2TAB2 utility could be used to perform this task.

Inspect file *ibound_zones.ndt*. This is also a node data table file. However, this file features both IBOUND and zonation columns. *ibound_zones.ndt* is easily constructed from *ibound.ndt* and *zones.ndt* through cutting and pasting using a text editor that supports block editing. (Most good editors do this.) *ibound_zones.ndt* also contains a final column in which layer numbers are ascribed to nodes. (This column can be pasted from the second part of the grid specification file *example.gsf*.) Layer numbers will shortly come in handy for viewing nodes pertaining to individual layers in PARAVIEW.

2.3 Using PARAVIEW for Display

PARAVIEW is a powerful public domain visualization utility. It can read many types of file to obtain the information which it plots, including a so-called “legacy vtk” file. We will use the USG2VTK utility from the PEST Groundwater Utilities Suite to write such a file. Note that this program is useable only where a MODFLOW-USG model has rectangular cells. (As we shall see shortly, the model grid which is the subject of the present tutorial is quadpatch-refined in areas of interest; nevertheless, all cells are still rectangular.) However, if a MODFLOW-USG model had been built using a package such as ALGOMESH which employs polygonal cells instead of rectangular cells, then USG2VTK1 should be used instead of USG2VTK for building a legacy VTK file.

Run USG2VTK, responding to its prompts as follows.

```
Enter name of MODFLOW-USG grid or CLN specification file: example.gsf

Enter name for VTK output file: example.vtk

Record scalar data in VTK file? [y/n]: y
Enter name of tabular file from which to read node data: ibound_zones.ndt

Enter column number of data to read (<Enter> if no more): 2
Is this integer or real data? [i/r]: i

Enter column number of data to read (<Enter> if no more): 3
Is this integer or real data? [i/r]: i

Enter column number of data to read (<Enter> if no more): 4
Is this integer or real data? [i/r]: i

Enter column number of data to read (<Enter> if no more): <Enter>
```

Now, if you have PARAVIEW installed on your computer, read file *example.vtk* and produce a display like the one shown in Figure 1. (The vertical exaggeration is 20. Cell selection is for IBOUND greater than zero, and colouring is according to zone.)

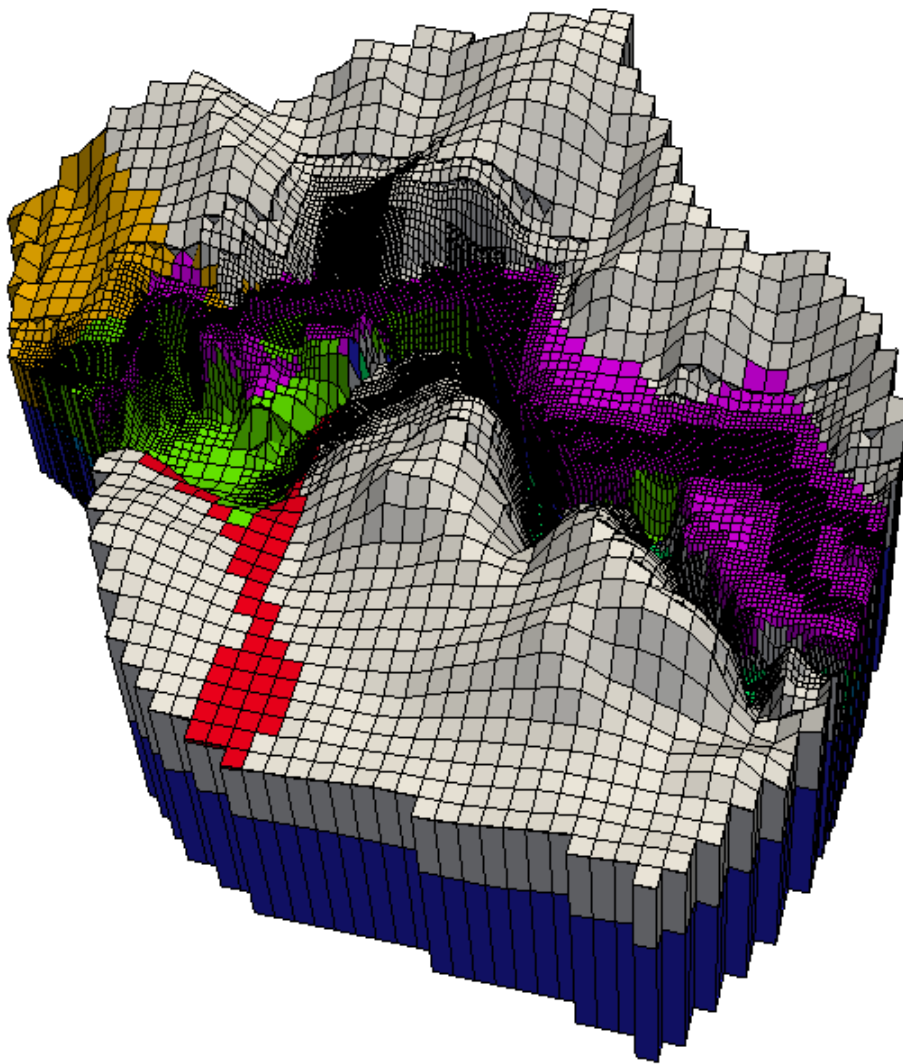


Figure 1. The model visualized in PARAVIEW.

Other programs provided with the Groundwater Utilities allow you to display a model in platforms other than PARAVIEW. USGNDTF2MIF allows you to write MIF/MID files that can be imported into a GIS. So does USGGRIDLAY. The latter program also allows you to write BLN files that can be readily imported into SURFER. We will do this now, writing a BLN file which depicts active cells in layer 4. Run USGGRIDLAY, responding to its prompts as follows.

```
Enter name of unstructured grid specification file: example.gsf

Enter layer number of interest: 4

Enter filename base for MIF/MID files (<Enter> if none): <Enter>
Enter filename base for BLN file (<Enter> if none): example.bln
Enter name for nodal xyz file (<Enter> if none): nodes.dat

Filter output using column of node data table file? [y/n]: y
Enter name of node data table file: ibound_zones.ndt
Enter column header: ibound
Does column contain integer or real data? [i/r]: i
Enter value for node omission: 0
```


When running the above program, we also write a file named *nodes.dat* which lists all active nodes in layer 4 together with their x, y and z coordinates. Have a look at this file if you are interested. If you have SURFER installed on your computer, start it up and import file *example.blm*. Figure 2 shows what you will see. As is apparent from the above series of prompts, BLN file construction could have been based on individual zones rather than on the whole model grid in any layer of interest. SURFER could then have been used to display cells in individual zones.



Figure 2. Model grid in layer 4 displayed by SURFER.

2.4 A Closer Look at Model Zonation

Figures 3a-d show the model, looking from the west, as displayed by PARAVIEW. In part (b) of this figure layer 1 is removed; in part (c) layers 1 and 2 are removed, while in part (d) layers 1 to 3 are removed. Cells are coloured according to zone. It is apparent that some zones occur in multiple model layers while some are restricted to an individual layer. Shortly we will parameterize the model using pilot points. These will be assigned to zones, not layers. The fact that zones are not layer-specific needs to be accommodated when interpolating from pilot points to the MODFLOW-USG grid.

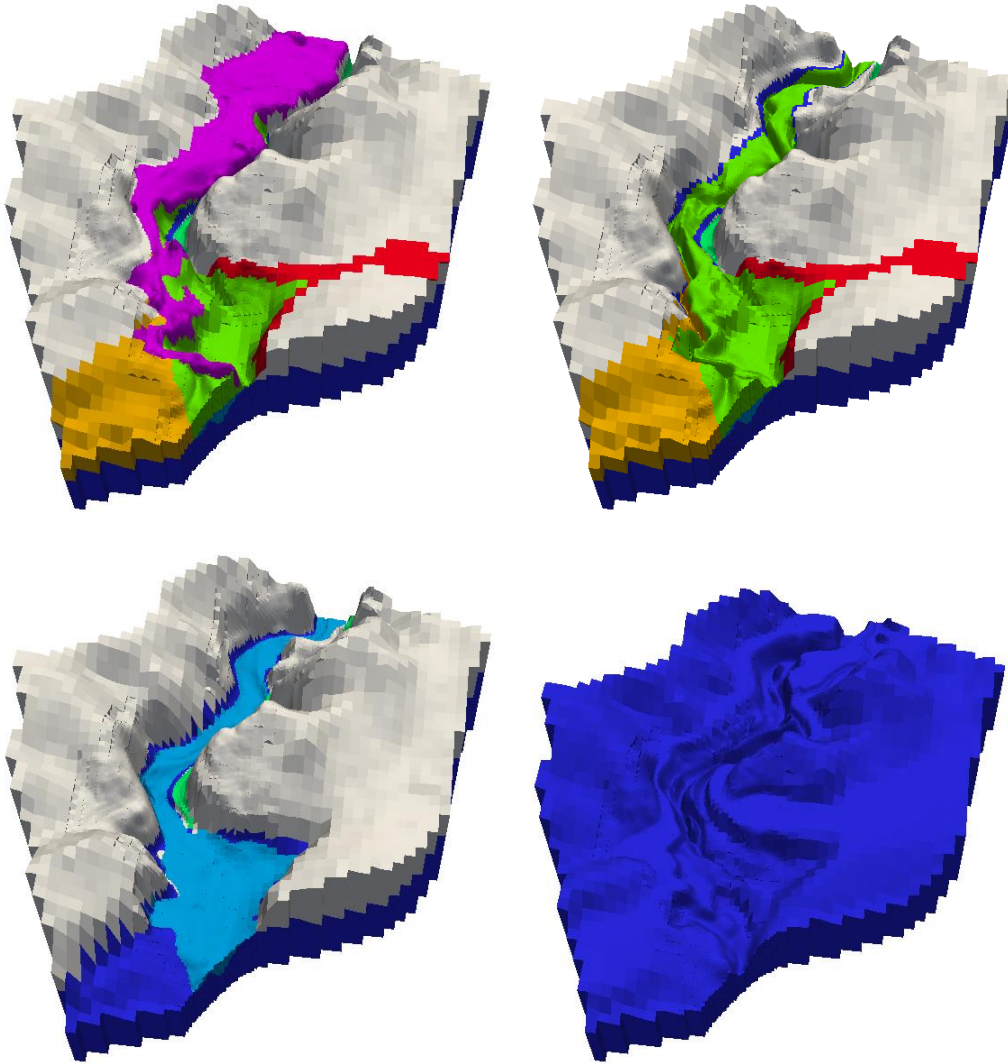


Figure 3. Top row left: layers 1 to 4. Top row right; layers 2 to 4. Bottom row left: layers 3 and 4; bottom row right: layer 4. Colouring is by zone.

In what follows, we will demonstrate PLPROC by using it to parameterize zones 3 and 5 of the model. Zone 3 is the deep blue-coloured zone that occupies the entirety of layer 4. Zone 5 is the light blue coloured zone that occupies the bottom of the river channel in layer 3.

3. PLPROC

3.1 Reading Model Grid Specifications

We will start slowly.

Inspect file *plproc1.in*. PLPROC undertakes complex tasks using a series of function calls. In a PLPROC input file comments begin with the “#” character. These lines are not executed; they are only used for explanation. The character “&” at the end of a line indicates continuation of that line to the next line.

The first thing we will do is read the MODFLOW-USG grid specification file *example.gsf*. In reading this file, PLPROC creates a CLIST named *cl_m* (for “CLIST model”). Once it has constructed this CLIST, PLPROC knows the x, y and z coordinates of every node of the MODFLOW-USG grid. At the same time, it creates a *cl_m*-dependent SLIST named *layer* (node layer numbers are listed in the second part of the MODFLOW-USG grid specification file). So PLPROC also knows the layer to which each node of the MODFLOW-USG grid belongs. However, in the present script it does not use this information.

Here is the function call.

```
cl_m = read_mf_usg_grid_specs(file=example.gsf, slist=layer)
```

A MODFLOW-USG grid specification file does not specify zonation, as this is unrelated to grid geometry; different zonations can be used on different occasions. To obtain the zonation used as a basis for hydraulic property assignment in the present case, PLPROC reads the *ibound_zones.ndt* node data table file which was discussed above. As is apparent from the second function call appearing in file *plproc1.in* (shown below), PLPROC reads zone numbers from the third column of this file and assigns them to an SLIST named *zone*. The parent CLIST for this *zone* SLIST is, once again, the *cl_m* CLIST which contains grid node coordinates.

```
read_list_file(reference_clist='cl_m', skiplines=1,      &
               slist=zone; column=3,                  &
               file='ibound_zones.ndt')
```

Then follow some reporting functions. These can be used in a PLPROC script to ensure that the script is doing what it is supposed to be doing. Using the following function calls, all entities generated by PLPROC are recorded in file *report1.dat*, while the coordinates of all points that comprise the *cl_m* CLIST, and the values of all SLISTS which have this CLIST as its parent, are reported in file *report2.dat*.

```
report_all_entities(file=report1.dat)
cl_m.report_dependent_lists(file='report2.dat')
```

Run PLPROC using the following command. (Substitute “plproc32” for “plproc64” if your machine does not have a 64-bit operating system.)

```
plproc64 plproc1.in
```

Now inspect files *report1.dat* and *report2.dat*.

3.2 Pilot Points

We are now going to parameterize each of zones 3 to 10 of our model using pilot points. Files containing the coordinates of pilot points, as well as hydraulic conductivity values associated with pilot points, are provided as files *pp_k_z[3-10].pts*. In this file-naming convention, “pp” stands for “pilot point”, “k” stands for hydraulic conductivity and “z” stands for “zone”. Where PEST is used for model calibration, a template for each of these files would normally be constructed so that PEST could transfer current hydraulic conductivity values to each of them prior to running the model. PLPROC would then be run in response to a command issued in the batch file which PEST runs as “the model”.

PLPROC refers to files such as *pp_k_z[3-10].pts* as “list files”. Data residing in list files is arranged in columns; the first column contains point identifiers, the second and third columns contain point x any y coordinates, the fourth column contains z coordinates (but only if points are three-dimensional), and subsequent columns contain real or integer values associated with listed points. More than one property or integer value can be associated with each point; hence the file read by PLPROC can have many columns.

Figure 4a shows pilot points that are used to inform zone 3 of our model while figure 4b shows pilot points that are used to inform zone 5.

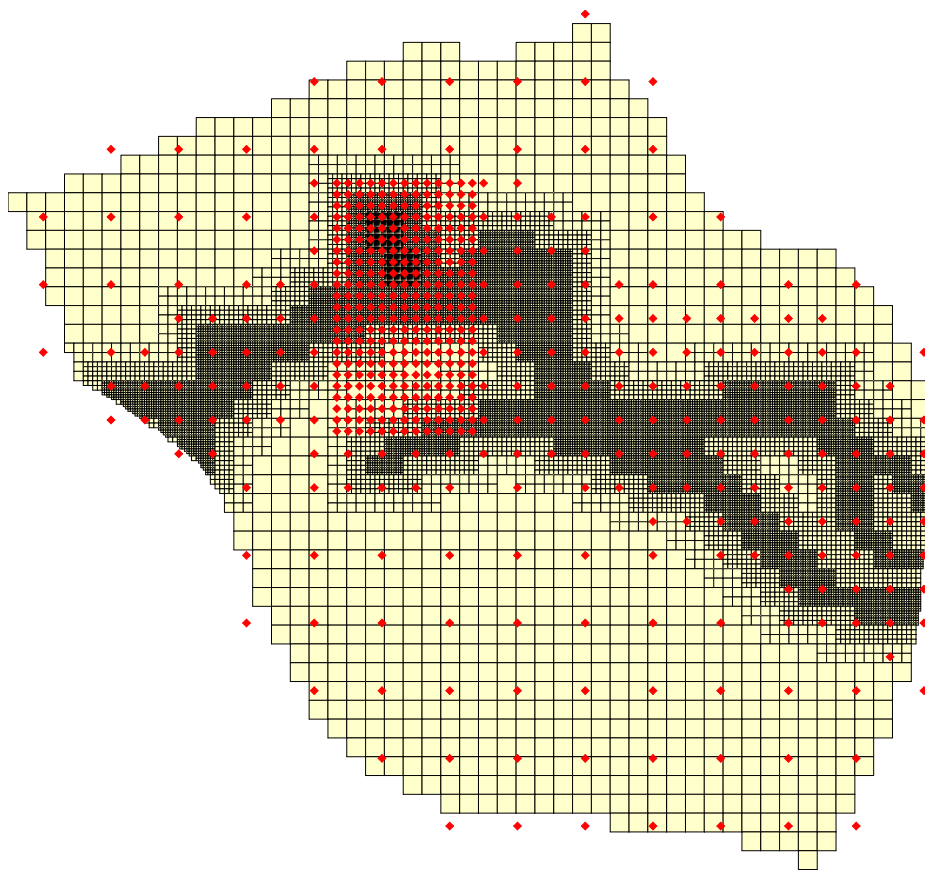


Figure 4a. Pilot points used to inform zone 3, together with model cells which comprise zone 3.

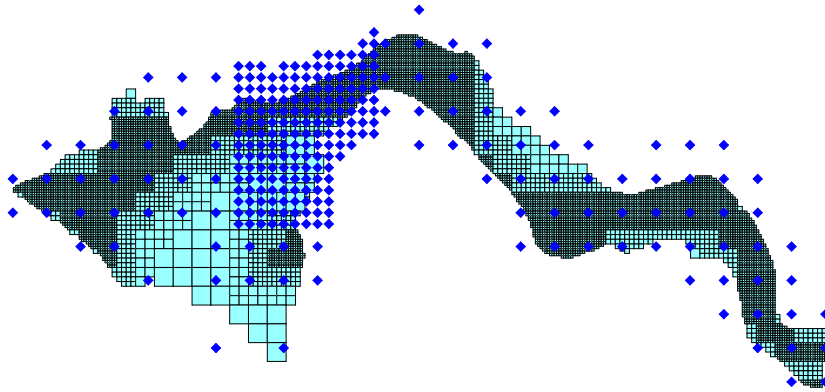


Figure 4b. Pilot points used to inform zone 5, together with model cells which comprise zone 5.

Inspect file *plproc2.in*. The first part of this file is the same as *plproc1.in* except that the reporting function calls have been moved to the bottom of the file. The places where these functions used to be listed are now occupied by a set of calls to function *read_list_file()*. One of these function calls is reproduced below.

```
c1_k3 = read_list_file(skiplines=1, dimensions=2,    &
                      plist='pp_k3'; column=4,      &
                      id_type='integer',            &
                      file='pp_k_z3.pts')
```

The above *read_list_file()* function call instructs PLPROC to read file *pp_k_z3.pts* containing data for pilot points assigned to zone 3. This function knows that the x and y coordinates of these points reside in the second and third columns of the file. These coordinates are used to create a new CLIST, in this case named *c1_k3*. A *c1_k3*-dependent PLIST named *pp_k3* is created at the same time. The elements of this PLIST contain hydraulic conductivities assigned to pilot points as read from the fourth column of file *pp_k_z3.pts*. If desired in later PLPROC processing, individual pilot points can be identified through the integer value comprising the first column of this file; this is established using the *id_type='integer'* argument.

The PLPROC script contained in file *plproc2.in* finishes with reporting functions; as stated above, these can provide assurance that PLPROC is working as expected. Once again, a record is made of all entities that PLPROC has in storage; these are written to file *record1.dat*. Meanwhile file *record2.dat* is used to store all data pertaining to the *c1_k3* CLIST and its dependent PLISTs (in this case the *pp_k3* PLIST).

Run PLPROC using the command

```
plproc64 plproc2.in
```

Inspect the contents of files *record1.dat* and *record2.dat*.

3.3 Interpolation from Pilot Points to Model Grid Nodes

Inspect file *plproc3.in*, which is a continuation of file *plproc2.in* but with the reporting functions moved to the end of the file. A call to function *new_plist()* marks the first part of the new section of this file. A new PLIST named *k_m* is created; its parent CLIST is *c1_m* (the CLIST that describes all nodes of the MODFLOW-USG model). This function assigns a default value of 0.0 to all nodes of the model grid. This PLIST will shortly house hydraulic conductivity values interpolated from pilot points.

```
k_m =new_plist(reference_clist=cl_m, value=0.0)
```

A series of calls to the radial basis interpolation function *rbf_interpolate_2d()* follows this. The first of this series is reproduced below.

```
k_m(select=(zone==3))=pp_k3.rbf_interpolate_2d(                                &
    transform='log',                                                            &
    rbf=imq; epsminsepfac=0.8,                                                &
    constant_term='yes',                                                       &
    report_file='report_k.dat')
```

The above function interpolates hydraulic conductivity values found in the *pp_k3* PLIST to elements of the *k_m* PLIST, but only if these elements are in zone 3. Actually, it interpolates the logs of hydraulic conductivities and then back-transforms to the domain of natural numbers after interpolation has taken place (this is effected using the *transform='log'* argument). Interpolation employs the inverse multiquadratic radial basis function type, this being described by the following equation.

$$f(r) = \frac{1}{\sqrt{1 + (\varepsilon r)^2}}$$

Using the *epsminsepfac=0.8* subargument of the *rbf* argument of the *rbf_interpolate_2d()* function, the ε parameter is decreed as being location-specific; for the radial basis function associated with any pilot point it is assigned a value equal to the reciprocal of 0.8 times the distance from that pilot point to its nearest neighbour.

Radial basis function interpolation requires that a high dimensional series of simultaneous equations be solved in order to evaluate coefficients associated with the functions. The numerical details of solving these equations are recorded in file *report_k.dat*. Such reporting is optional; it is effected using the *report_file='report_k.dat'* argument.

More reporting takes place at end of file *plproc_3.in*. Once again, all entities housed in PLPROC's memory are recorded in file *report1.dat*. Details of PLISTs and SLISTs associated with the *cl_m* CLIST are recorded in file *report2.dat*.

Run PLPROC using the command

```
plproc64 plproc3.in
```

When it has completed execution inspect files *report1.dat*, *report2.dat* and *report_k.dat*. Notice from *report2.dat* that interpolation has taken place to all active nodes of the MODFLOW-USG grid.

3.4 Evaluating Vertical Hydraulic Conductivities

The values ascribed to pilot points contained in files *pp_k_z[3-10].pts* are actually horizontal hydraulic conductivities. We will assume that vertical anisotropy is uniform in each zone. Values for zone-specific vertical anisotropy will be read from a file named *vanis.dat*. Normally a template would exist for this file so that these can be estimated by PEST. Inspect *vanis.dat*. It has two columns. Items in the first column are the names of so-called "scalar variables" (PLPROC nomenclature) that will be used to denote zonal vertical anisotropies; these names are retained by PLPROC. The second column lists values of these variables. This file is read using the following function call recorded in file *plproc4.in*.

```
read_scalar_file(file="vanis.dat", valuecolumn=2,      &
    namecolumn=1, skiplines=1)
```

Next a PLIST named *kz_m* is created. Its parent is the *cl_m* CLIST, so it has an element for every node in the MODFLOW-USG grid. All of these elements are given an initial value of zero.

```
kz_m=new_plist(reference_clist = cl_m, value=0.0)
```

Vertical hydraulic conductivity is next calculated for every active node of the MODFLOW-USG grid using a series of simple, zone-specific equations that involve the *k_m* PLIST and the respective zone-specific vertical anisotropy. The first such equation is shown below.

```
kz_m(select=(zone==3)) = k_m/vanis_z3
```

File *plproc4.in* ends with the usual reporting functions. Run PLPROC using the command

```
plproc64 plproc4.in
```

It is obvious from file *report2.dat* that values of vertical hydraulic conductivity have been assigned to all active cells of the MODFLOW-USG grid.

3.5 Writing Model Input Files

File *plproc5.in* includes the contents of file *plproc4.in* but adds the ability to write nodal horizontal and vertical hydraulic conductivity values to MODFLOW-USG input files. The final two function calls in file *plproc5.in* record horizontal and vertical hydraulic conductivities to files named *mfusg_k.dat* and *mfusg_kz.dat* respectively. These files are opened and assigned unit numbers in the MODFLOW-USG name file; see file *example.nam* – part of the MODFLOW-USG input dataset. Meanwhile the unit numbers associated with these files are referenced in the MODFLOW-USG LPF file so that MODFLOW-USG knows that it must read horizontal and vertical hydraulic conductivities from them; see file *example.lpf*.

Like PEST, PLPROC uses template files to transfer the numbers that it generates to model input files. However, template files used by PLPROC can be much more sophisticated than those used by PEST because they can contain embedded PLPROC functions. Below is the embedded function contained in file *mfusg_k.tpl* which is used to write the MODFLOW-USG input file *mfusg_k.dat*. (In the present case the embedded function constitutes the sole contents of the template file. In other cases, just as for a PEST template file, the PLPROC template file can contain many other numbers and strings; these are transferred directly to the model input file which PLPROC writes.)

```
$#p k_m.write_in_sequence(format="(1x,1pg14.7) ")
```

The “\$#p” string at the beginning of the above line informs PLPROC that a function call follows and that the contents of this line are not therefore meant to be directly transferred to the model input file. Instead, PLPROC is informed that it must write the contents of the *k_m* PLIST to the model input file, with one element of the PLIST recorded on each line of that file. The name of the model input file is provided in the *write_model_input_file()* function recorded in file *plproc5.in* along with the name of the template file that is used to write this file. The function call is as follows.

```
write_model_input_file(template_file = 'mfusg_k.tpl',                                &  
                        model_input_file = 'mfusg_k.dat')
```

Run PLPROC using the command

```
plproc64 plproc5.in
```

Inspect files *mfusg_k.dat* and *mfusg_kz.dat*. These files are ready for use by MODFLOW-USG.

4. DISPLAYING IN PARAVIEW

4.1 Hydraulic Property Data

Inspect file *ibound_properties.ndt*. This file is easily built by pasting the single column of data appearing in each of files *mfusg_k.dat* and *mfusg_kz.dat* into file *ibound_zones.ndt* and adding a header to these columns. We will now build a legacy VTK file which contains all of these data. This will allow us to visualize the interpolated hydraulic conductivity fields in three-dimensions.

Run USG2VTK, responding to its prompts as follows. (Note that, as for any other member of the Groundwater Utilities Suite, if you make a mistake when entering data you can backtrack to the previous prompt by responding to the current prompt with “e” – for “escape” – followed by <Enter>.)

```
Enter name of MODFLOW-USG grid or CLN specification file: example.gsf

Enter name for VTK output file: example1.vtk

Record scalar data in VTK file? [y/n]: y
Enter name of tabular file from which to read node data: ibound_properties.ndt

Enter column number of data to read (<Enter> if no more): 2
Is this integer or real data? [i/r]: i

Enter column number of data to read (<Enter> if no more): 3
Is this integer or real data? [i/r]: i

Enter column number of data to read (<Enter> if no more): 4
Is this integer or real data? [i/r]: i

Enter column number of data to read (<Enter> if no more): 5
Is this integer or real data? [i/r]: r

Enter column number of data to read (<Enter> if no more): 6
Is this integer or real data? [i/r]: r

Enter column number of data to read (<Enter> if no more): <Enter>
```

If you have PARAVIEW installed on your computer, read file *example1.vtk* and colour according to KH (after selecting only cells with a non-zero IBOUND value for display). Use a logarithmic scale for colouring. Figure 5 shows the results, for all layers and then with layers successively removed.

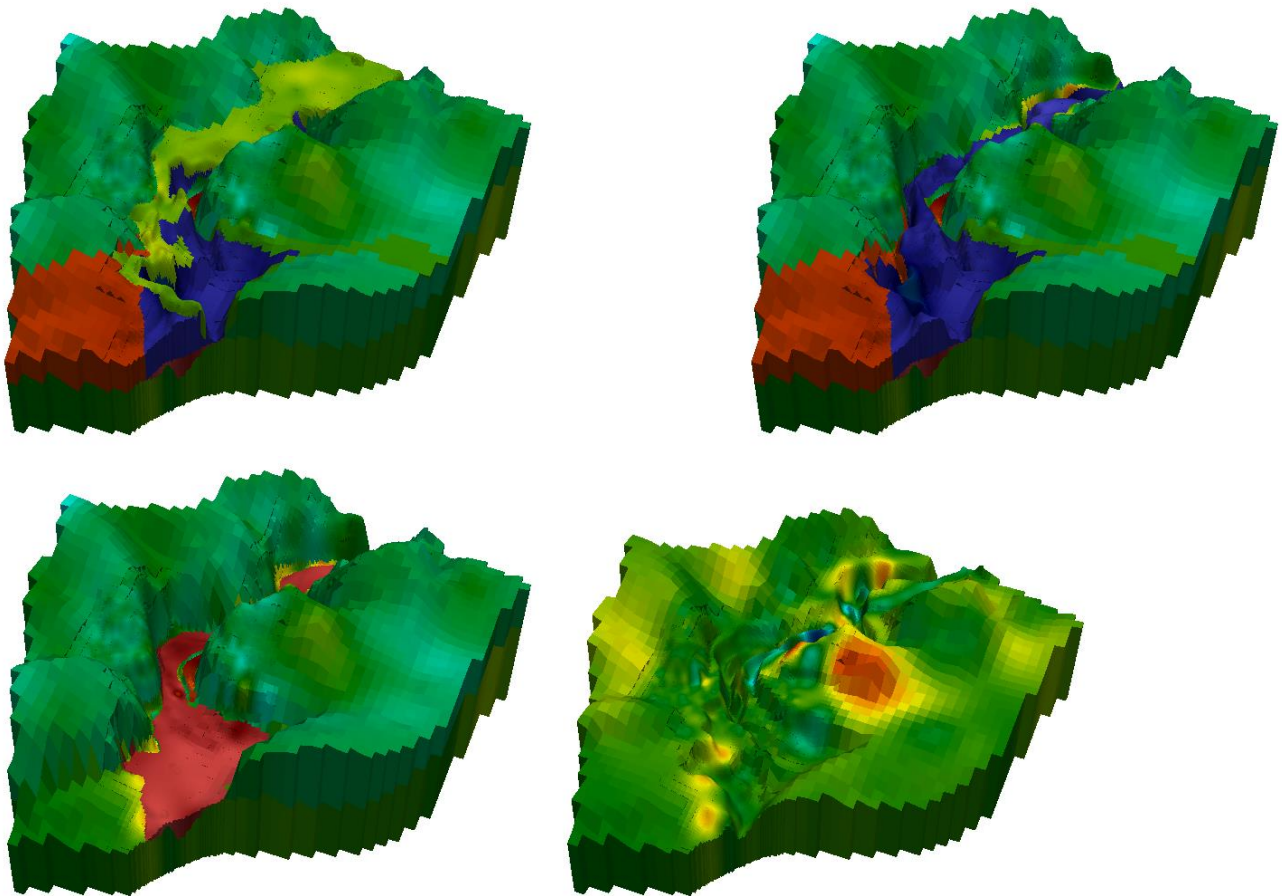


Figure 5. Top row left: layers 1 to 4. Top row right; layers 2 to 4. Bottom row left: layers 3 and 4; bottom row right: layer 4. Colouring is by log of horizontal hydraulic conductivity.

4.2 Adding Pilot Points to the Picture

Just for the sake of creating some eye-candy, we will now visualize horizontal hydraulic conductivity in zone 5 only, together with pilot points associated with that zone. We will actually plot the pilot points in three dimensions. However before doing this we need to provide each pilot point with a vertical coordinate. This is accomplished using the USGADDZCOORD utility which interpolates from the MODFLOW-USG grid to the locations of pilot points. Run USGADDZCOORD, responding to its prompts as follows.

```
Enter name of MODFLOW-USG grid specification file: example.gsf
- file example.gsf read ok.

Enter name of activity node data table file (<Enter> if none): ibound.ndt
Enter column number containing activities: 2
- file ibound.ndt read ok.

Enter name of tabular data file: pp_k_z5.pts
Does this file have a header line? [y/n]: y
Enter column number containing layer data (<Enter> if none): <Enter>
Enter layer number to which this file pertains: 3

Enter power of inverse distance to use in interpolation: 2
Enter search radius: 1000
```

Enter maximum number of points to use in interpolation: **4**

Enter name for new tabular data file: **ppz_k_z5.txt**

Inspect file *ppz_k_z5.txt*. A column of z coordinates has been added.

Now import file *ppz_k_z5.txt* into PARAVIEW. The field delimiter is the space character; inform PARAVIEW to detect numeric columns, use string delimiters, save headers and merge consecutive delimiters. The data is imported as a spreadsheet. Now use the PARAVIEW “table to points” filter to create a set of three-dimensional points based on the pilot point data; X column is “EAST”, Y column is “NORTH” and Z column is “Z”. Check the “keep all arrays” box. Display the pilot points (don’t forget to alter the Z scale transformation factor to 20 in accordance with the scale factor used for plotting model grid data).

Figure 6a shows a view of zone 5, coloured according to the range of horizontal hydraulic conductivities appearing in that zone only (to a logarithmic scale). The blue “bull’s eye” suggests erroneous local data used in the calibration process and/or that regularisation is insufficient. The pilot points are added to this plot in Figure 6b, with hydraulic conductivities made transparent so that all of the pilot points are visible.

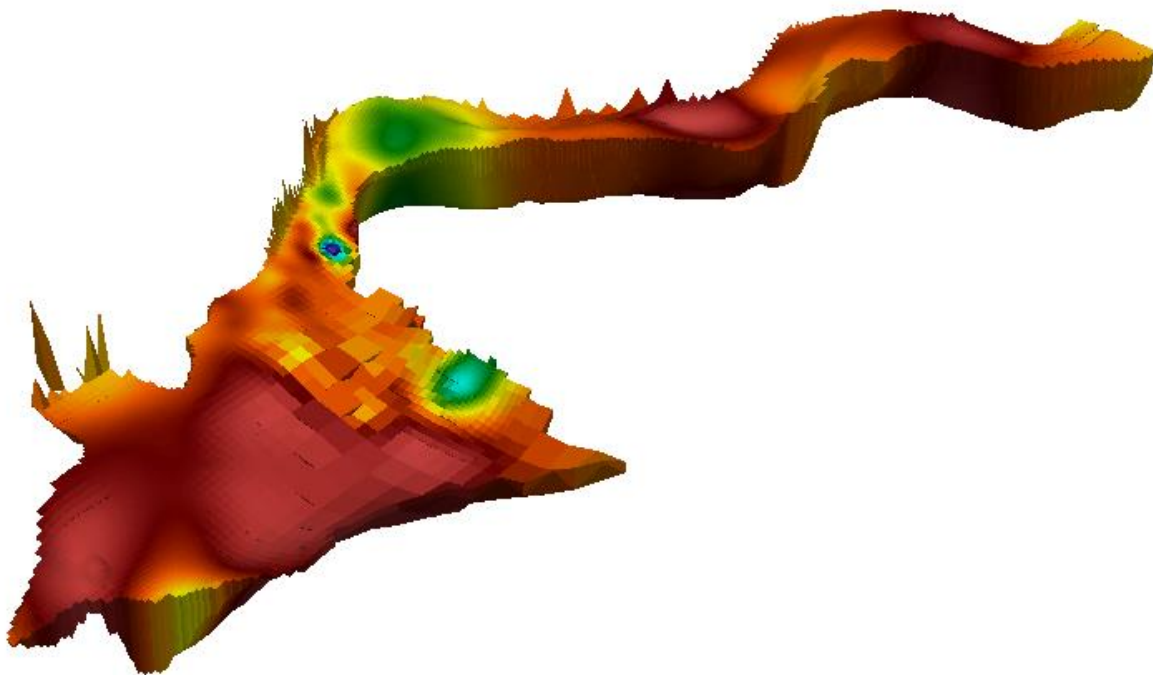


Figure 6a. Close-up view of zone 5, with the colour scale adjusted to represent the range of horizontal hydraulic conductivities featured in this unit.

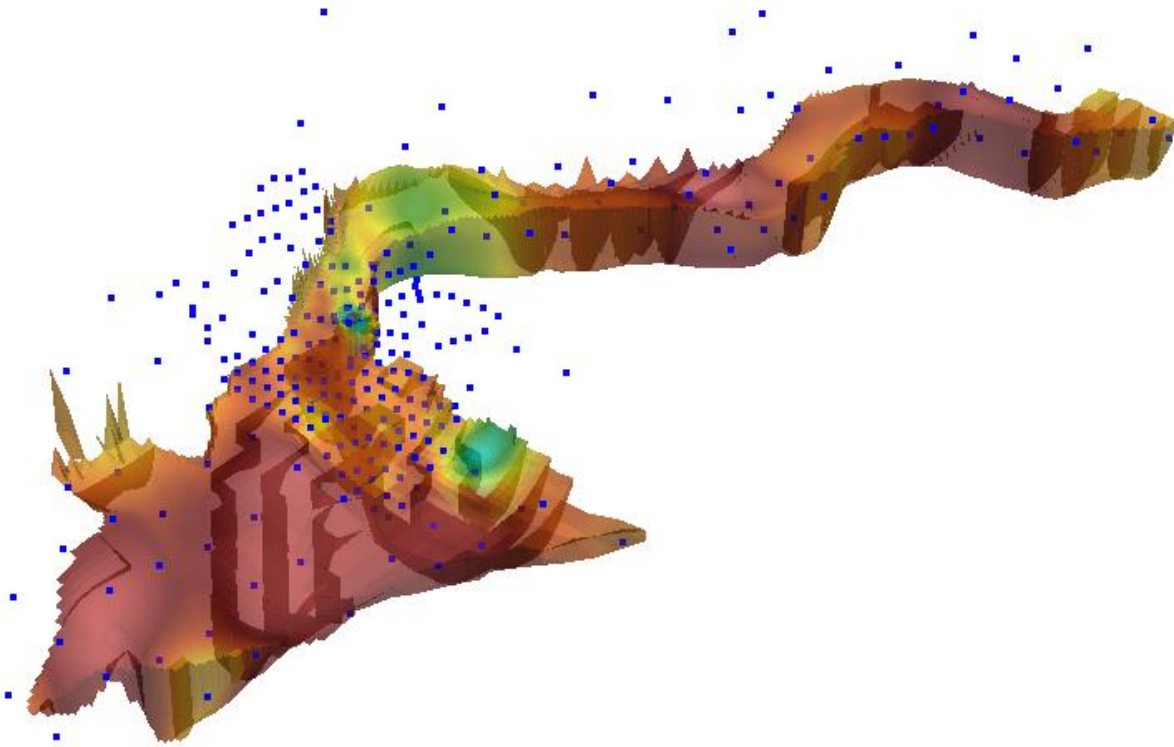


Figure 6b. Pilot points specific to zone 5 added to the plot of Figure 6a.



gmdsi.org

CRICOS NO 00114A

BHP



Flinders
UNIVERSITY



NATIONAL CENTRE FOR
GROUNDWATER
RESEARCH AND TRAINING

RioTinto