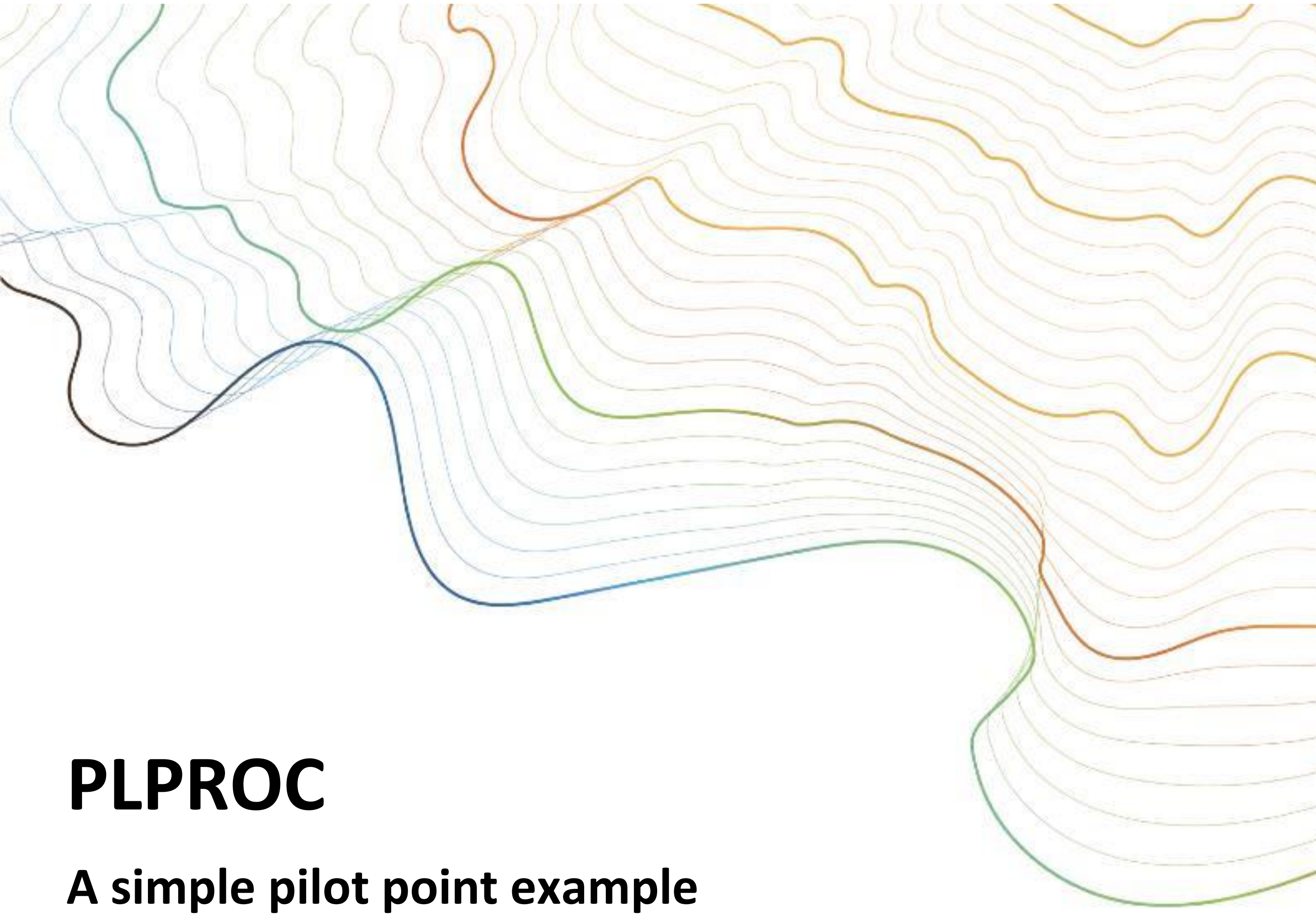

PLPROC

A simple pilot point example

A GMDSI tutorial

by Rui Hugman and John Doherty

gmdsi
Groundwater Modelling
Decision Support Initiative

BHP

Flinders UNIVERSITY

NATIONAL CENTRE FOR GROUNDWATER RESEARCH AND TRAINING

RioTinto

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.

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

“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 and quantify 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 6 model with multiple zones and layers using pilot points. This demonstration does not comprise a comprehensive review of all functionality available in PLPROC. Nor does it illustrate 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.

As well as PLPROC, this tutorial makes use of the MF62VTK utility from the PEST Groundwater Utilities Suite. 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 makes use of utilities that enable visualisation of properties and system states associated with a MODFLOW 6 model. The GMDSI Model Visualisation and Display tutorial demonstrates how to setup and use these utilities. The current tutorial assumes that the reader is familiar with their use.

2. BACKGROUND

This tutorial demonstrates the use of PLPROC to parameterize a version of the MODFLOW 6 model used in other GMDSI tutorials, using pilot points. This would commonly be integrated as part of a PEST/PEST++ workflow, in which PLPROC parameterizes a model from values determined by PEST/PEST++ prior to running the model. Although the workflow demonstrated here is specific to MODFLOW 6, the concepts can be easily adapted to other models and/or file structures. It is assumed that the reader is familiar with MODFLOW 6 file structures.

2.1 The Groundwater Model

Relevant model files are provided in the tutorial folder. These include the MODFLOW 6 binary discretisation file (*model.disv.grb*) which, as is discussed in documentation of MODFLOW 6, contains complete geometric specifications of the MODFLOW 6 model grid. It also stores model IDOMAIN values. In the present tutorial these have been used to assign cells to zones 1 to 4. PLPROC and the MF62VTK utilities can read this file to obtain information on the geometry of the model.

The model uses a rectangular quadtree-refined grid (created using USGS software GRIDGEN) to represent the 3 layered aquifer system displayed in Figure 1. Each layer has distinct hydraulic properties and patterns of spatial heterogeneity. The upper Layer 1 is a phreatic aquifer. Layer 2 is a low permeability aquitard, while layer 3 is a confined aquifer. Layer 1 is subdivided into two zones.

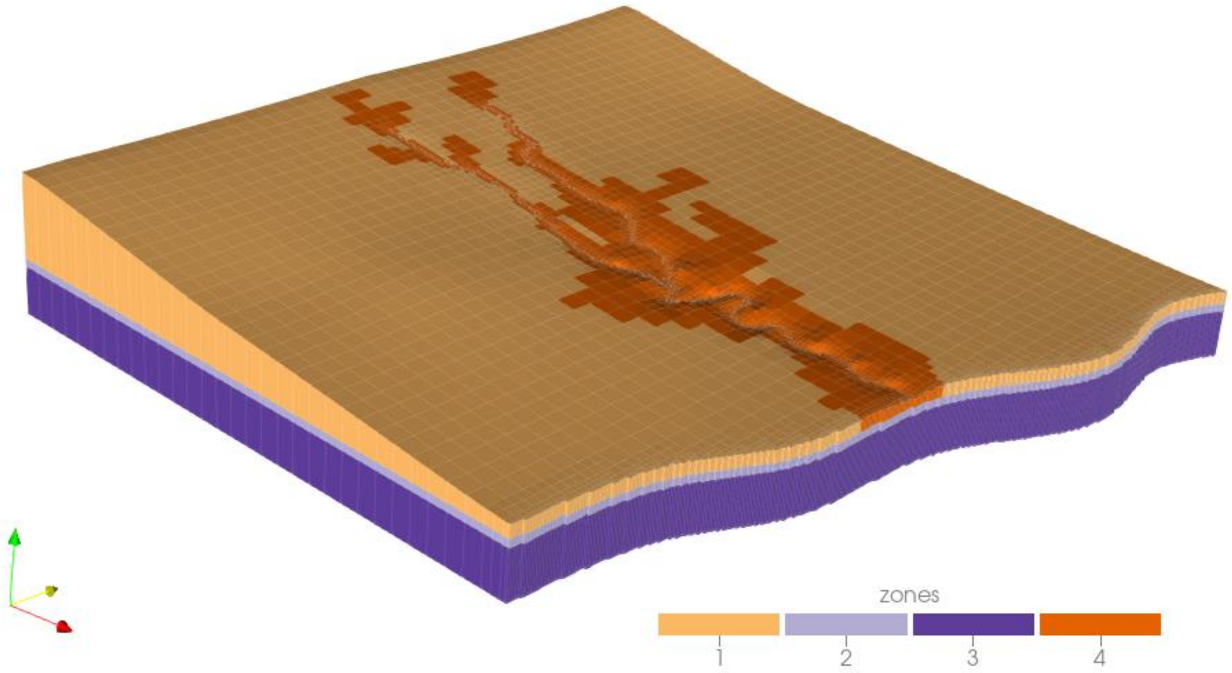

Figure 1 Three-dimensional view of the model grid; zones are defined by IDOMAIN value. Vertical scale is exaggerated x10.

PLPROC is used to assign values for hydraulic properties associated with pilot-points, to individual model cells belonging to each of the model layers and zones. Hydraulic properties that are of interest in the present case are hydraulic conductivity (K), specific yield (Sy) and specific storage (Ss). K is parameterized for each layer. Sy and Ss are only parameterized for the top and bottom layers, respectively. K and Sy in layer 1 are parameterized for two distinct zones (1 and 4).

External array files which house model hydraulic properties (*npf_k_layer[1-3].ref, sto_ss_layer3.ref* and *sto_sy_layer1.ref*) are also provided. These files are read by MODFLOW 6 to assign hydraulic properties to the model using its OPEN/CLOSE option. Open them in a text editor and take a look. They contain a single column of data listing hydraulic properties (in order of cell number) for all cells within the respective model layer.

A note on the MODFLOW OPEN/CLOSE File Protocol

MODFLOW provides the option to house parameter lists or arrays in external files. This provides several benefits, including the following.

- External software such as PPROC can easily re-write an array that is stored in such a file.
- The same file can be used to inform multiple layers or even properties in the same model.

When creating external model input files using MODFLOW's OPEN/CLOSE protocol, note the following.

- Inform MODFLOW 6 that it should read these external arrays using the (FREE) protocol. The reading of numbers then requires only that they be separated from each other by white space (including a new line); they are not required to occupy fixed positions on a line. However, values must still be listed in order of increasing cell index.
- It is good practice to record numbers one to a line in an external file. This practice is "safe", as it can be used for both structured and unstructured grids.

2.2 Viewing the Model

Visualizing the model is not a requirement for completion of this tutorial. However, it will allow you to see the effect of what you are doing. PARAVIEW is a powerful public domain visualization package. 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 MF62VTK utility from the PEST Groundwater Utilities Suite to write such a file. Other GMDSI tutorials demonstrate further use of this and other visualization-support utilities.

Run MF62VTK, responding to its prompts as follows. User inputs are indicated in ***bold italics***. Alternatively, simply run the *run_mf62vtk.bat* batch file provided in the tutorial folder.

```
Program MF62VTK writes a "legacy" VTK file based on a MODFLOW6 binary grid
   output file and, optionally, associated node data.

 Enter name of MODFLOW6 binary grid file: model.disv.grb
 - file model.disv.grb read ok.

 Enter name for VTK output file: model.vtk

 Record scalar data in VTK file?  [y/n]: y
 Obtain scalar data from tabular file or layer files? [t/l]: l

 Enter filename base for set 1: (<Enter> if no more): npf_k_layer
 Do these files hold integer or real data? [i/r]: r
 Enter filename extension: .ref
 Enter no-data value if file missing: -999
 Enter label for this data type: k
 - file npf_k_layer1.ref read ok.
 - file npf_k_layer2.ref read ok.
 - file npf_k_layer3.ref read ok.
```

```
Enter filename base for set 2: (<Enter> if no more): sto_ss_layer
Do these files hold integer or real data? [i/r]: r
Enter filename extension: .ref
Enter no-data value if file missing: -999
Enter label for this data type: ss
- no-data value of -999 used for layer 1 as file sto_ss_layer1.ref not found
- no-data value of -999 used for layer 2 as file sto_ss_layer2.ref not found
- file sto_ss_layer3.ref read ok.

Enter filename base for set 3: (<Enter> if no more): sto_sy_layer
Do these files hold integer or real data? [i/r]: r
Enter filename extension: .ref
Enter no-data value if file missing: -999
Enter label for this data type: sy
- file sto_sy_layer1.ref read ok.
- no-data value of -999 used for layer 2 as file sto_sy_layer2.ref not found
- no-data value of -999 used for layer 3 as file sto_sy_layer3.ref not found

Enter filename base for set 4: (<Enter> if no more): <Enter>
- file model0.vtk written ok.
```

Open *model.vtk* in PARAVIEW to visualize the model grid and hydraulic properties. You are now able to display the grid, zones (i.e. IDOMAIN values) and initial hydraulic property values. You will note that the latter are currently uniform for each zone, as is displayed in Figure 2.

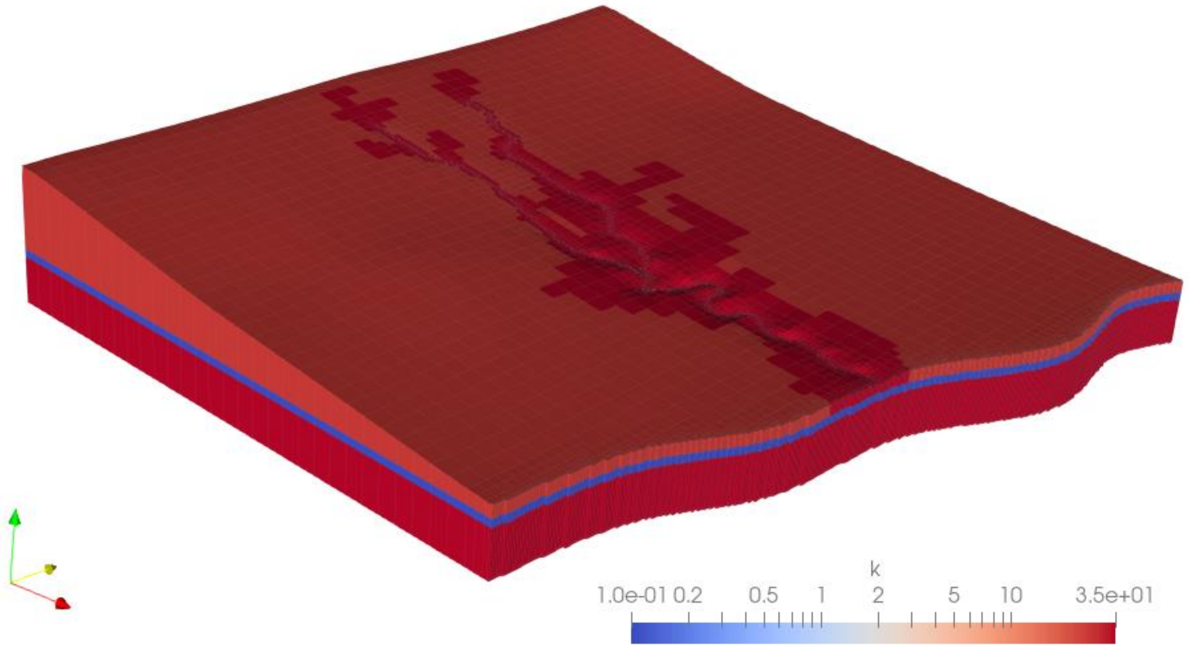

Figure 2 Model grid and initial uniform values of hydraulic conductivity within each zone. Vertical scale is exaggerated x10.

All files generated so far can be found in the folder *t1*.

2.3 The Pilot Points

Pilot point identifiers, coordinates, and associated hydraulic property values are provided in file *ppoints.dat*. Where PEST is used for model calibration, a template of such a file would normally be constructed so that PEST could transfer current parameter values to each pilot point prior to running the model. Under these circumstances, PLPROC would be run using a command issued in a batch file which PEST runs as “the model”.

PLPROC refers to files such as *ppoints.dat* as “list files”. Data residing in list files are arranged in columns. The first column contains point identifiers, the second and third columns contain point *x* and *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 3 shows the pilot points used to inform zone 1 and zone 4 in layer 1. Zones 2 and 3 (which represent layer 2 and 3 respectively) are informed by both sets of pilot points. Pilot point locations used here have been selected manually. Their placement was roughly in accordance with the following principles:

(a) Make sure that placement of pilot points extends out to the model boundaries (albeit with a lower spatial density);
(b) Place them between observation wells and (i) other observation wells in the up or down-gradient direction (ii) pumping wells and (ii) outflow/inflow boundaries;
(c) Employ a greater spatial density of pilot points where there is greater density of observation points;
(d) Preserve a minimum spatial density of pilot points everywhere else.

Other GMDSI tutorials may expand upon strategies for pilot point placement.

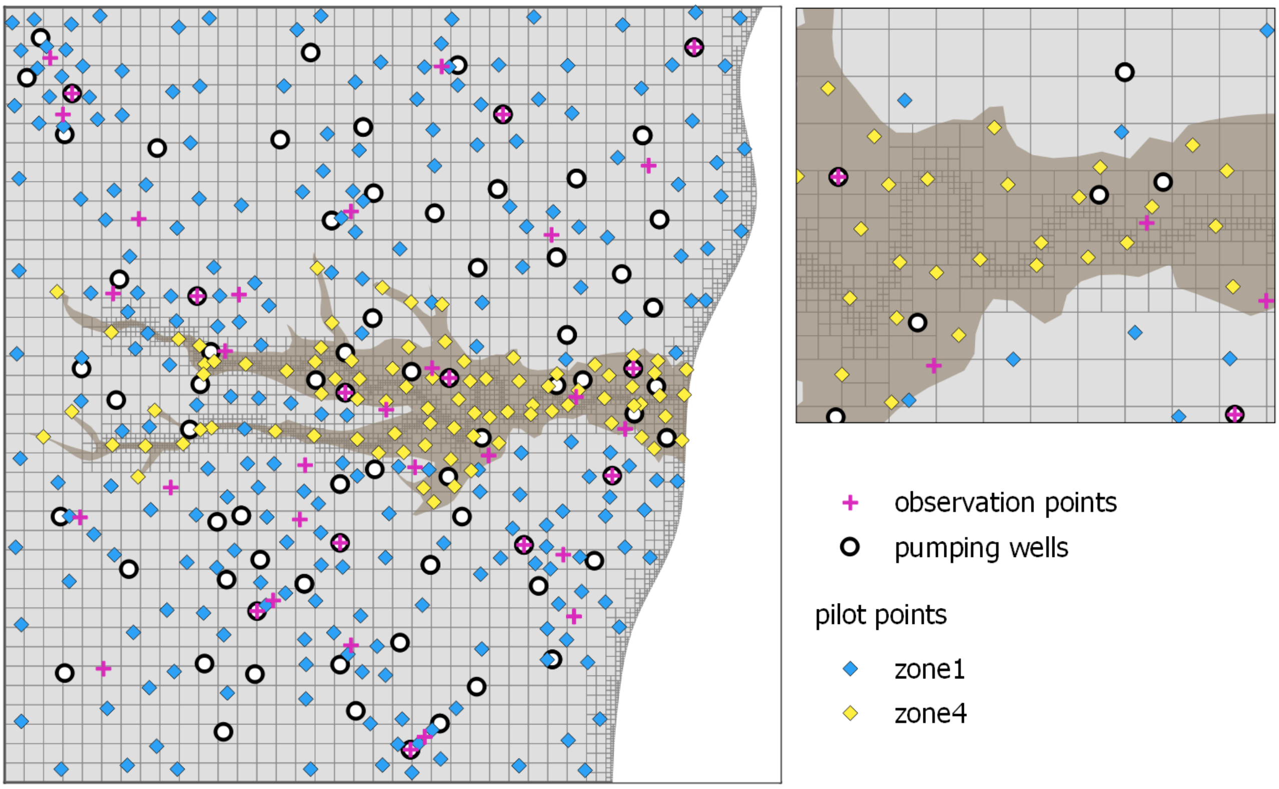

Figure 3 Model grid and locations of observation points, pumping wells and pilot points. All of these pilot points are used in both of layers 2 and 3 (i.e. in zones 2 and 3).

In some instances, if different pilot points are used for different zones, it may be more practical to create separate list files for each set of pilot points. However, in this tutorial it is convenient to use a single file.

3. PLPROC

A PLPROC user invokes various aspects of PLPROC functionality by providing it with a text input file containing a set of statements or commands which tell it what to do next. Each of these statements calls a function or embodies an equation; some processing control functionality is also available. If a function is called, the specifics of the action invoked by the function are conveyed to PLPROC through the function's arguments. The syntax employed for function calls resembles that of programming languages with which many PLPROC users will be familiar. A sequence of PLPROC commands provided in a single such input file can thus be considered to be a "program" or "script"; the latter terminology will be employed herein. 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 following section of the tutorial will demonstrate how to set up a PLPROC input file to parameterize the model described above using pilot points. It is advisable to have the PLPROC manual close at hand. There are five main steps:

1. Read the model geometry;
2. Read the pilot point geometry;
3. Define how to interpolate from pilot points to model cells;
4. Undertake the interpolation;
5. Write hydraulic property values to model files using a PLPROC template file.

3.1 Reading Model Grid Specifications

The first thing that we must do is instruct PLPROC to read the MODFLOW 6 binary grid file *model.disv.grb*. In reading this file, PLPROC creates a CLIST (a “coordinate” list). Once it has constructed this CLIST, PLPROC knows the *x*, *y* and *z* (only if parameterisation is three-dimensional) coordinates of every node of the MODFLOW 6 grid.

At the same time, CLIST-dependent SLIST’s (“selection” lists) can be created. These allow us to ascribe nodes to layers and/or zones. There are several ways to do this, depending on whether parameterisation is two- or three-dimensional. (See the PLPROC manual for details).

Let us get started.

PLPROC has a function for reading MODFLOW 6 binary grid files; this is named *read_mf6_grid_specs()*. (Other PLPROC functions read other model grid types).

1 Open an empty text file. Name it *plproc1.dat* and type in the function below:

```
cl_mf = read_mf6_grid_specs(file=model.disv.grb,                     &
                            dimensions=2,                            &
                            slist_layer_idomain=idomain1;  layer=1,   &
                            slist_layer_idomain=idomain2;  layer=2,   &
                            slist_layer_idomain=idomain3;  layer=3)
```

2 This function tells PLPROC to create a CLIST named *cl_mf*, using the binary grid file *model.disv.grb*.

3 As our parametrisation is two-dimensional (interpolation from pilot points to the model grid does not involve the *z* dimension), the function specifies *dimensions=2*. In this case, each element of the CLIST is ascribed only an *x* and a *y* coordinate; these are independent of any model layer.

4 Lastly, three SLISTs are associated with the new CLIST; we name these *idomain1*, *idomain2* and *idomain3*. They contain the IDOMAIN value of every cell in a specified model layer. These will come into play later on.

3.2 Reporting Functions

Before continuing any further, let us check if everything is working correctly by adding 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.

5 Add the following function to the bottom of the *plproc1.dat* script.

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

6 Using this function call, all entities generated by PLPROC are recorded in file *report1.dat*, while the coordinates of all points that comprise the *cl_mf* CLIST, and the values of all SLISTS which have this CLIST as its parent, are reported in file *report2.dat*.

7 Go ahead and save the file. You can now run PLPROC.

8 Open the command line in your working folder. Type *plproc plproc.dat* and press <enter>. If all went well, you should see the following.

```
PLPROC Version 3.00. Watermark Numerical Computing.

Reading and storing contents of PLPROC script file plproc1.dat...
Processing commands in PLPROC script file...

> cl_mf=read_mf6_grid_specs(file=model.disv.grb,dimensions=2,slist_laye...
> report_all_entities(file=report1.dat)
> cl_mf.report_dependent_lists(file='report2.dat')

End of file: no more commands to process.
```

9 Go ahead and inspect files *report1.dat* and *report2.dat* using any text editor.

3.3 Reading Pilot Points

Next, we need to instruct PLPROC to read the list file *ppoints.dat*, which contains pilot point data. We accomplish this using the *read_list_file()* function. As previously described, file *ppoints.dat* contains the location and hydraulic property values associated with each pilot point.

10 Open *ppoints.dat* in a text editor or spreadsheet and take a look.

11 Now, add the following function to the *plproc1.dat* script. Place it above the reporting functions. (Henceforth, the reporting function shall always be kept at the end of the script file.)

```
cl_pp = read_list_file(file=ppoints.dat,                     &
                        id_type=character,                   &
                        dimensions=2,                        &
                        skiplines=1,                         &
                        slist=zone;    column=4,             &
                        plist=kh1_pp;   column=5,            &
                        plist=sy1_pp;   column=6,            &
                        plist=kh2_pp;   column=7,            &
                        plist=kh3_pp;   column=8,            &
                        plist=ss3_pp;   column=9)
```

12 The above *read_list_file()* function call instructs PLPROC to read file *ppoints.dat* containing data for pilot points. 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 two-dimensional CLIST, in this case named *cl_pp*. (Note that entries in file *ppoints.dat* are both tab and space delimited. PLPROC allows either. Tab delimiters can make a file look untidy when viewed in a text editor. PLPROC does not object to multiple spaces/tabs between numbers; however, it does not tolerate missing numbers. Use of multiple spaces instead of tabs allows file like this to be tidied up if you would like this.)

13 As *ppoints.dat* contains column headers, PLPROC is instructed to skip the first line when reading this file; this is done using the "*skiplines=1*" argument.

14 An SLIST named *zone* is created using values read from column 4 of the file. If you check the file, you will see that this column contains zone numbers. Note that it only contains the numbers 1 and 4. This is because all the pilot points have been assigned to zones 1 and 4.

15 At the same time, *cl_ pp* dependent PLISTs are created for each of the hydraulic properties listed in *ppoints.dat*. These are named *kh1_pp, sy1_pp* and so on. PLPROC is instructed to read the

pertinent column in file *ppoints.dat* in order to assign values to the respective PLIST. For example, a PLIST named *kh1_pp* is created using the values in column 5 using the argument: “*plist=kh1_pp; column=5*”

16 If desired in later PLPROC processing, individual pilot points can be identified through the pilot point name comprising the first column of this file. Pilot points can be identified by position in the list; user-supplied integers, or user-supplied character strings. In this case the character string options is adopted. This is established using the “*id_type=character”* argument.

17 Run PLPROC again and inspect files *report1.dat* and *report2.dat.*

3.4 Calculate Interpolation Factors

Next, we define how values are interpolated from pilot points to model grid nodes. PLPROC has several options to carry out spatial interpolation. Here we use the *calc_kriging_factors_auto_2d()* and the *krige_using_file()* functions. This is a two-step process. First, *calc_kriging_factors_auto_2d()* writes a file which links elements in a source CLIST (i.e. *cl_mf*) to elements in a target CLIST (i.e. *cl_pp*). Then, *krige_using_file()* carries out the interpolation to calculate element values of one PLIST from those of another PLIST.

Recall that we are working in two-dimensions. Therefore, the *x* and *y* coordinates of our model nodes are the same for each layer. Furthermore, we are using the same pilot points for each layer; so, their *x* and *y* coordinates are also the same. However, in layer 1 we have two distinct zones, that is zone 1 and zone 4).

Because interpolation between pilot points and model nodes in layer 2 is the same as that in layer 3, the same kriging factors can be used for interpolation to cells in each of these layers. However, this is not the case for layer 1, as it contains two zones. Because each zone has different nodes and pilot points, a different set of kriging factors must be calculated for each zone.

Let us start with the kriging factors for layers 2 and 3.

18 Add the following function to the *plproc1.dat* script.

```
calc_kriging_factors_auto_2d(                                  &
          target_clist=cl_mf;select=(idomain3.ne.0),           &
          source_clist=cl_pp,                                  &
          file=factors23.dat)
```

19 The first argument in the function (*target_clist=cl_mf;select=(idomain3.ne.0)*) instructs PLPROC to interpolate from pilot points to a subset of elements in *cl_mf* (the model grid CLIST). This subset is comprised of those for which the IDOMAIN value is “not equal” to zero. (The string “!=” could be used instead of “.ne.”) These are identified in the idomain3 SLIST. (In this case, we could also use *idomain1* or *idomain2* as our selector SLISTS as they all have IDOMAIN=0 in the same cells.) All this selection sub-argument is doing, is constraining the interpolation to cells that are active (i.e. IDOMAIN ≠ 0) in the MODFLOW 6 model. This is not essential, as no damage is done when interpolation takes place to cells that are not used by the model. However, if there are many such cells then calculation of kriging factors can take time. Also, the file that is used to store interpolation factors can become unnecessarily large.)

20 The second argument defines the source CLIST as the pilot point CLIST *cl_pp*. No selection is defined, meaning that interpolation can take place from all pilot points to the model grid.

21 Lastly, PLPROC is instructed to save the kriging factors that it calculates in an external file named *factors23.dat*. This will allow us to re-use these factors without having to re-calculate them on each occasion that we need to undertake interpolation from pilot points to the model grid (which is very often, when PEST is used to estimate the values that are assigned to pilot points).

Now let us define the function for zone1 in layer 1.

22 Add the following function to the *plproc1.dat* script.

```
calc_kriging_factors_auto_2d(                                 &
           target_clist=cl_mf;select=(idomain1==1),           &
           source_clist=cl_pp;select=(zone==1),               &
           file=factors1.dat)
```

23 Here, the target CLIST has a different selection argument. Recall in step 4 that IDOMAIN values in layer 1 were ascribed to elements of an SLIST named *idomain1*. In the current function, the target CLIST elements are defined as only those which have an *idomain1* value equal to 1 (*idomain1==1*). These are model cells which belong to zone 1. (Note that ".EQ." could be used instead of "==".) Thus, interpolation takes place from pilot points that are assigned to zone 1, to model cells that are assigned to zone 1. It is the user's responsibility to ensure that pilot point emplacement is appropriate for this.

24 Next, the source CLIST (the pilot point CLIST) is also subject to a selection argument. In this case we only want to interpolate from pilot points that are designates as belonging to zone 1. Therefore, only elements from the source CLIST that have a *zone* SLIST value equal to 1 are used.

25 Factors are saved in the file *factors1.dat.*

The function for zone 4 in layer 1 is similar.

26 Add the following function to the *plproc1.dat* script.

```
calc_kriging_factors_auto_2d(                                 &
           target_clist=cl_mf;select=(idomain1==4),           &
           source_clist=cl_pp;select=(Zone==4),               &
           file=factors4.dat)
```

27 In this case, the selection arguments for target and source CLISTs specify *idomain1* and *zone* SLIST values that are equal to 4.

28 Factors are saved in the file *factors4.dat.*

29 Go ahead and run PLPROC again. You should now see the three interpolation factor files that it creates in your working folder. Open them in a text editor and take a look if you wish.

3.5 Interpolation from Pilot Points to Model Grid Nodes

So far, we have only instructed PLPROC to calculate interpolation factors. A nice thing about kriging is that the interpolation process is independent of the values that are actually interpolated. Hence interpolation factors can be calculated once, and then used forever more. So finally, we instruct PLPROC to interpolate values from pilot point PLISTs to model PLISTs, and then to record the latter in model input files. We accomplish this using the *krige_using_file()* and *write_model_input_file()* functions.

30 Start off by creating a dummy PLIST associated with the model CLIST *cl_mf*. Add the following function to the *plproc1.dat* script.

```
  prop_mf=new_plist(reference_clist=cl_mf,value=1.0)
```

31 This function simply creates a PLIST named *prop_mf* from the model CLIST and assigns the value 1.0 to all of its elements. This PLIST is going to serve as a "template" PLIST to make repetitive use of subsequent functions somewhat easier.

Now we proceed to interpolate from each pilot point PLIST to the *prop_mf* PLIST, and then record the interpolated values in an external file ready for reading by MODFLOW 6. Let us start with K in zone 1.

32 Add the following function to the *plproc1.dat* script.

```
  prop_mf=kh1_pp.krige_using_file(file='factors1.dat',transform='log')
```

33 This function updates the *prop_mf* PLIST (associated with model nodes), by interpolating values from the *kh1_pp* PLIST (associated with pilot points) using the interpolation factors housed in file *factors1.dat*. Recall that these are the interpolation factors for zone 1. Only elements in *prop_mf* that correspond to model nodes within zone 1 will be updated.

34 Actually, the above function interpolates the logs of values and then back-transforms to the domain of natural numbers after interpolation has taken place (this is effected using the *transform='log'* argument). This is appropriate for interpolating quantities like hydraulic conductivity, storage coefficient and specific yield whose values can span a very large range.

Before writing values to a layer 1 model input file, we need to include the K values for zone 4 in layer 1 in the *prop_mf* PLIST.

35 Add the following function to the *plproc1.dat* script.

```
  prop_mf=kh1_pp.krige_using_file(file='factors4.dat',transform='log')
```

36 The only difference is that the factor file has been changed to *factors4.dat*, which houses interpolation factors for zone 4.

We can now need a function that allows us to write values recorded in the *prop_mf* PLIST to a model input file. 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. These allow the transfer of all or part of a PLIST to a model input file, without the need to individually identify each element of the PLIST. Let us create a template file now.

37 Open a new blank text file and add the function below. Save this file as *gen_mf_array.tpl.* (Note that in this case, the only contents of the file which PLPROC will write are elements of a PLPROC PLIST. Hence the template of this file contains only a single embedded function. In general, a template file contains the entire contents of a model input file, together with one or more embedded functions. PLPROC inserts the contents of its respective PLISTs at places identified by the embedded functions. The way in which these contents are recorded is also specified in the embedded function.)

```
$#p prop_mf.write_in_sequence(format="(1x,1pg18.11)")
```

38 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 *prop_mf* 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 which we will now add to the *plproc1.dat* script.

39 Add the following function to the *plproc1.dat* script.

```
write_model_input_file(template_file=gen_mf_array.tpl,                  &
                       model_input_file=npf_k_layer1.ref)
```

40 This function instructs PLPROC to use *gen_mf_array.tpl* (which we just created) as a template file to write a model input file named *npf_k_layer1.ref*.

41 Go ahead and run PLPROC again. You should now see a new file named *npf_k_layer1.ref*. Open it in a text editor and take a look. You should see a single column of values with 5011 rows. These are the values of K for every cell in layer 1 of the model.

42 Great! So now all that we need to do is write model input files for all of the other hydraulic properties. Add the following functions to the *plproc1.dat* script.

```
### sy1 ###
prop_mf=0.25
prop_mf=sy1_pp.krige_using_file(file='factors1.dat',transform='log')
prop_mf=sy1_pp.krige_using_file(file='factors4.dat',transform='log')
write_model_input_file(template_file=gen_mf_array.tpl,                  &
                       model_input_file=sto_sy_layer1.ref)

### kh2 ###
prop_mf=1.0E-2
prop_mf=kh2_pp.krige_using_file(file='factors23.dat',transform='log')
write_model_input_file(template_file=gen_mf_array.tpl,                  &
                       model_input_file=npf_k_layer2.ref)

### kh3 ###
prop_mf=0.5
prop_mf=kh3_pp.krige_using_file(file='factors23.dat',transform='log')
write_model_input_file(template_file=gen_mf_array.tpl,                  &
                       model_input_file=npf_k_layer3.ref)

### ss3 ###
prop_mf=0.5
prop_mf=ss3_pp.krige_using_file(file='factors23.dat',transform='log')
write_model_input_file(template_file=gen_mf_array.tpl,                  &
                       model_input_file=sto_ss_layer3.ref)
```

43 Note that the factor file varies according to which zone to which hydraulic properties belong.

44 Run PLPROC one last time. You should now see have five **.ref* files in your working folder.

3.6 View the Model Again

Repeat the steps in Section 2.2 (or simply execute the *run_mf62vtk.bat* batch file) and reload file *model.vtk* file in PARAVIEW. Display the values for each of the hydraulic properties. Note how they have changed.

All files generated so far can be found in the folder *t2*.

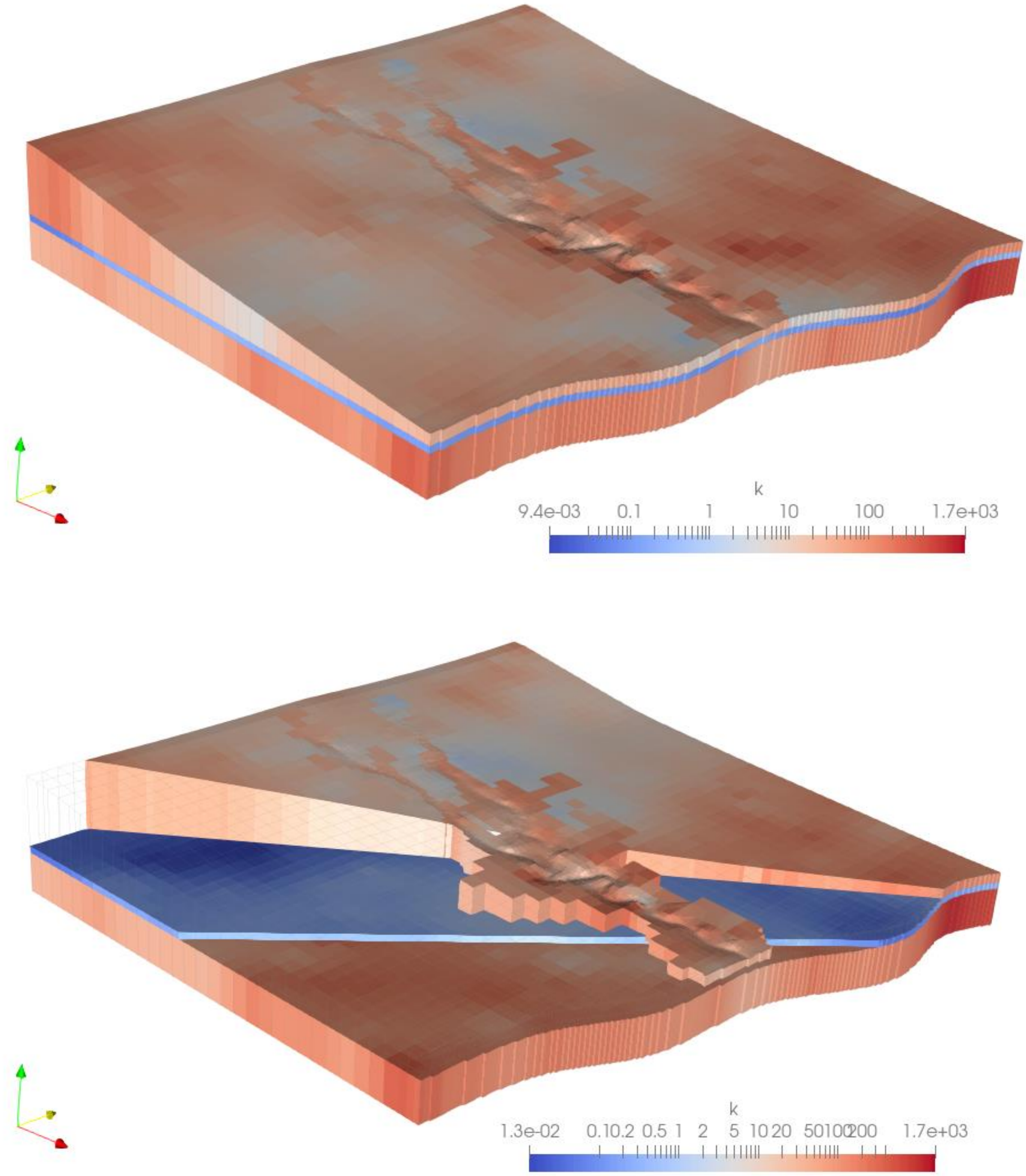

Figure 4 Model grid and spatial distribution of hydraulic conductivities interpolated from pilot points to the model grid using PLPROC. Vertical scale is exaggerated x10.

4. VARIATIONS

Because PLPROC is model-independent, it can be used in conjunction with any model. However certain of its functions are specific to various members of the MODFLOW family. These were written to simplify use of PLPROC with MODFLOW.

Function *read_mf_grid_specs()* reads a grid specification file for a traditional, structured-grid, version of MODFLOW. This grid specification file allows construction of two-dimensional CLISTs, and easily-arranged interpolation of pilot points to these CLISTs. Three-dimensional interpolation from pilot points to grids of a traditional MODFLOW model can also be undertaken but requires a little more effort to set up.

In contrast, setting up two- or three-dimensional pilot point interpolation to cells of MODFLOW 6 and MODFLOW-USG models is an easy matter. A MODFLOW-USG grid specification file can be read using function *read_mfusg_grid_specs()*. Prior to three-dimensional MODFLOW 6 interpolation, the *read_mf6_grid_specs()* function can be used, but in a slightly different way from that illustrated above.

When working with traditional, unstructured-grid, MODFLOW models, PLPROC can read integer and real arrays from dedicated array files that can also be read by MODFLOW itself. Integer arrays can be used for selection of grid subareas for special processing, as was demonstrated above. Real arrays can hold an existing set of layer-specific hydraulic properties which PLPROC can modify. It can then record modified properties in a new real array for the use of MODFLOW.

PLPROC can undertake mathematical operations of arbitrary complexity between PLISTS that represent either pilot points or the cells of a model grid. Furthermore, it is not necessary for pilot point parameters to represent hydraulic properties; they can represent multipliers of existing hydraulic properties. They can multiply another set of pilot point parameters; the resulting pilot-point-based PLIST can then be interpolated to a model grid. Alternatively (and often more conveniently), multiplier parameters that are associated with pilot points can be interpolated to a model grid where parameter multiplication can take place on a model-cell-by-model-cell basis.

PLPROC offers a number of specialized functions for spatial interpolation from pilot points to a model grid. Kriging based on a spatially-varying variogram is demonstrated above; it is implement using the *calc_kriging_factors_auto_2d()* function. Another function allows anisotropy of interpolation to vary with the direction of alluvial boundaries. Radial basis functions and inverse-power-of-distance constitute other interpolation options.

Still other PLPROC functions are better tuned to interpolation down linear features such as rivers and streams. These are demonstrated in another tutorial.

gmdsi

Groundwater Modelling
Decision Support Initiative

gmdsi.org

BHP

Flinders
UNIVERSITY

NATIONAL CENTRE FOR
GROUNDWATER
RESEARCH AND TRAINING

RioTinto