



SEGLISTS

Interpolation along linear features

A GMDSI tutorial

by Rui Hugman and John Doherty



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.....	5
2. Background.....	6
2.1 The Groundwater Model	6
2.2 Viewing the Model.....	7
3. Interpolating to Linear Features	9
3.1 The Easy Way (Using Pilot Points)	9
The Pilot Points.....	9
The PLPROC Input File	10
3.2 The Slightly Harder Way (Using Pilot Points and Zones).....	15
3.3 The Truly Linear Way (Using SEGLISTS)	20
Building the SEGLIST	20
Generating the Pilot Points	22

1. INTRODUCTION

This tutorial demonstrates several options for spatial parameterization of linear and polyline features. In a groundwater model, these may represent entities such as streams, rivers, faults, or fracture networks. The hydraulic properties of these features may vary along their lengths. If this variability is relevant for a prediction of interest, then its representation in a model is important.

Pilot points comprise a useful two- or three-dimensional spatial parameterization device. This tutorial demonstrates how they can also be used in parameterization of one-dimensional entities that lie within the two- or three-dimensional domain of a groundwater model. Piecewise linear interpolation from pilot points to these entities is implemented using PLPROC – a PEST utility support program. In this tutorial PLPROC is used in conjunction with MODFLOW 6.

“PLPROC” stands for “Parameter List PROCessor”. PLPROC is designed to facilitate history-matching of large numerical models by serving as a model-independent pre-processor for such models. PLPROC allows a modeller to 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 two GMDSI tutorials [PLPROC: Basics](#) and [PLPROC: a simple pilot point example](#) provide a gentle introduction to PLPROC. The current tutorial assumes some familiarity with PLPROC. This can be gained by completed these tutorials.

As well as PLPROC, this tutorial makes use of the MF62GIS utility from the PEST Groundwater Utilities Suite. Executable versions of all programs belonging to the PEST [Groundwater Utilities](#) 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 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, you should have [QGIS](#) (or a GIS package with similar functionality) installed on your computer. QGIS is open source and freely available through the world wide web. The present tutorial makes use of utilities that enable visualisation of properties 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 you are already familiar with their use.

2. BACKGROUND

This tutorial demonstrates the use of PLPROC in parameterizing a polylinear feature in a variant of a MODFLOW 6 model that is used in other GMDSI tutorials as well. The procedures described herein would normally be implemented as part of a PEST/PEST++ workflow, in which PLPROC parameterizes a model that is undergoing history-matching under the control of PEST or PEST++. Although the workflow described herein is specific to MODFLOW 6, the concepts that it demonstrates can be easily adapted to other models and/or file structures. It is assumed that the reader is familiar with MODFLOW 6 file structures.

Several folders are provided. These are tutorial checkpoints. Each contains the files that should have been produced by the end of each of the following chapters. Should you encounter any problems, they may be useful in troubleshooting. They also allow you to jump into the tutorial at any stage. However, it is recommended that you at least read through those parts of the tutorial that you do not complete.

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 contains complete geometric specifications of the MODFLOW 6 model grid; see documentation of MODFLOW 6 for full details. PLPROC, as well as utilities such as MF62GIS from the PEST Groundwater Utilities Suite can read this file to obtain information on the geometry of the model. A plan view of the model grid is displayed in Figure 1.

Streams that transect the model domain are represented using MODFLOW drain (i.e. DRN) boundary conditions. The current tutorial demonstrates several methods for parameterizing the conductances of drain cells. PLPROC assigns conductance values to DRN boundaries in individual model cells based on those associated with pilot-points.

In the MODFLOW 6 input dataset that is provided with this tutorial, the external array file *model.drn_stress_period_data_1.txt*, houses DRN hydraulic properties. This file is read by MODFLOW 6 using its OPEN/CLOSE option. Open it in a text editor and take a look.

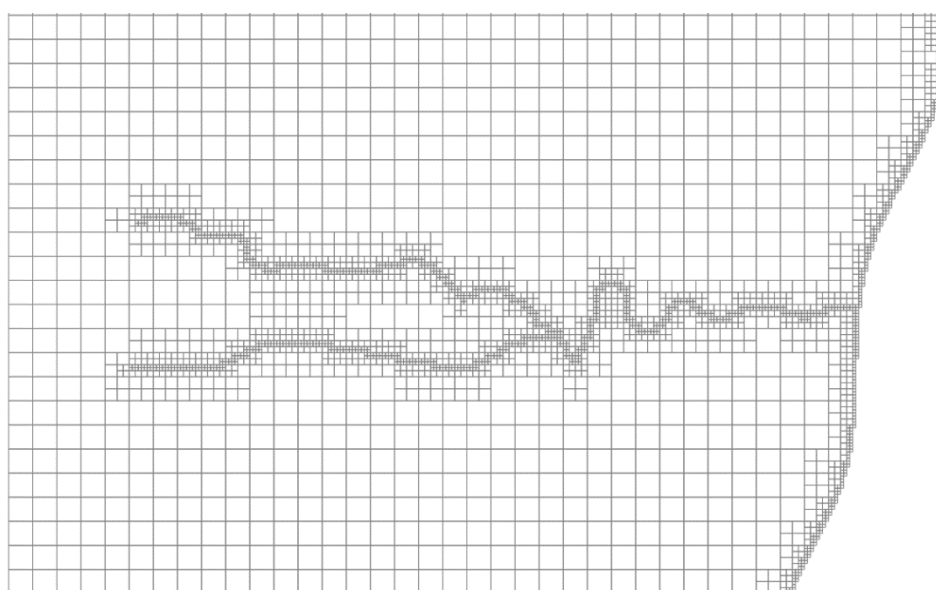


Figure 1 Plan view of the model grid. The grid is refined along the stream network, as well as along the right-hand model boundary.

A note on the MODFLOW OPEN/CLOSE File Protocol

MODFLOW provides the option to store lists or arrays of model input data in external files. This provides several benefits, including the following.

- External software such as PPROC can easily replace or modify an array or list that is stored in such a file.
- The same file can be reused to inform multiple layers or stress periods in the same model.

When creating text files that will be read using MODFLOW's OPEN/CLOSE protocol, note the following.

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

That being said, the methods described in this tutorial do not depend on data being supplied in an external file. PLPROC can read, manipulate, and alter data lists even in a standard MODFLOW package input file (see pertinent PLPROC functions). However, for simplicity and coherency with other GMDSI tutorials, the external file method is employed here.

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 (and is a little less boring than simply manipulating numbers). We will use the MF62GIS utility from the PEST Groundwater Utilities Suite to write MIF/MID files of the model grid. These can be read by most GIS software (including QGIS and SURFER). The GMDSI [Model Visualisation and Display](#) tutorial demonstrates further use of this and other visualization-support utilities. If you are comfortable using Python, similar functionality is available through Flopy.

As stated above, specifications of model DRN boundary conditions are stored in a file named *model.drn_stress_period_data_1.txt* (provided in the tutorial folder). This file is read by MODFLOW using its OPEN/CLOSE protocol. However, the MF62GIS utility expects a slightly different file structure. We need to add column headers, remove the layer number column, and ensure all columns are integers or real numbers.

- 1 Make a copy of *model.drn_stress_period_data_1.txt*. Name it *drains.dat*.
- 2 Open *drains.dat* in a text editor or spreadsheet software. Delete the first column. Then, insert a line at the top of the file and provide column headers for each of the remaining columns. Name the columns: *icpl*, *elevation*, *conductance*, and *reach*.
- 3 Lastly, MF62GIS requires that all columns contain integers or real numbers. The *reach* column currently contains text. Remove the text "stream-" from all rows.
- 4 The first few rows of *drains.dat* should now look like this:

icpl	elevation	conductance	reach
2830	140.94892883	1000.00000000	1
2842	139.41760254	1000.00000000	1
2843	138.98307800	1000.00000000	1
2845	139.76692200	1000.00000000	1

Run MF62GIS, responding to its prompts as follows. User inputs are indicated in ***bold italics***. Alternatively, simply run the *run_mf62gis.bat* batch file provided in the tutorial folder.

Program MF62GIS writes a BLN file and MIF/MID files for a MODFLOW6 model.

```
Enter name of MODFLOW6 binary grid file: model.disv.grb
- file model.disv.grb read ok.
Enter layer number of interest: 1
Enter name for BLN file (<Enter> if none): <enter>
Enter filename base for MIF/MID files (<Enter> if none): drains
Enter name of tabular data file (<Enter> if none): drains.dat
The following columns have been detected in the table file.
Indicate whether you would like pertinent data transferred to MIF/MID file.
  Data in column labelled "elevation"? [y/n]: n

  Data in column labelled "conductance"? [y/n]: y
  Does column contain integer or real data? [i/r]: r
  Enter value for missing data: 0
  Add or replace values for duplicated cells [a/r]: r

  Data in column labelled "reach"? [y/n]: y
  Does column contain integer or real data? [i/r]: i
  Enter value for missing data: 0
  Add or replace values for duplicated cells [a/r]: r

- reading tabular data file...
- 553 lines of data read from file drains.dat.
- file drains.mif written ok.
- file drains.mid written ok.
```

Open the MID/MIF file in your GIS software of choice to visualize the model grid and drain properties. Figure 2 displays values for conductance and reach number. As you can see, reaches are numbered 1 to 3 while a single conductance value is currently assigned to all DRN cells representing streams. The following chapters describe how to assign spatially varying conductances using PLPROC. Several methods will be demonstrated. In all cases this will be done in a way that allows these conductances to be adjusted using PEST or PEST++. At the end of each chapter, you can apply the same steps as those outlined above to visualize the outcomes of each method.

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

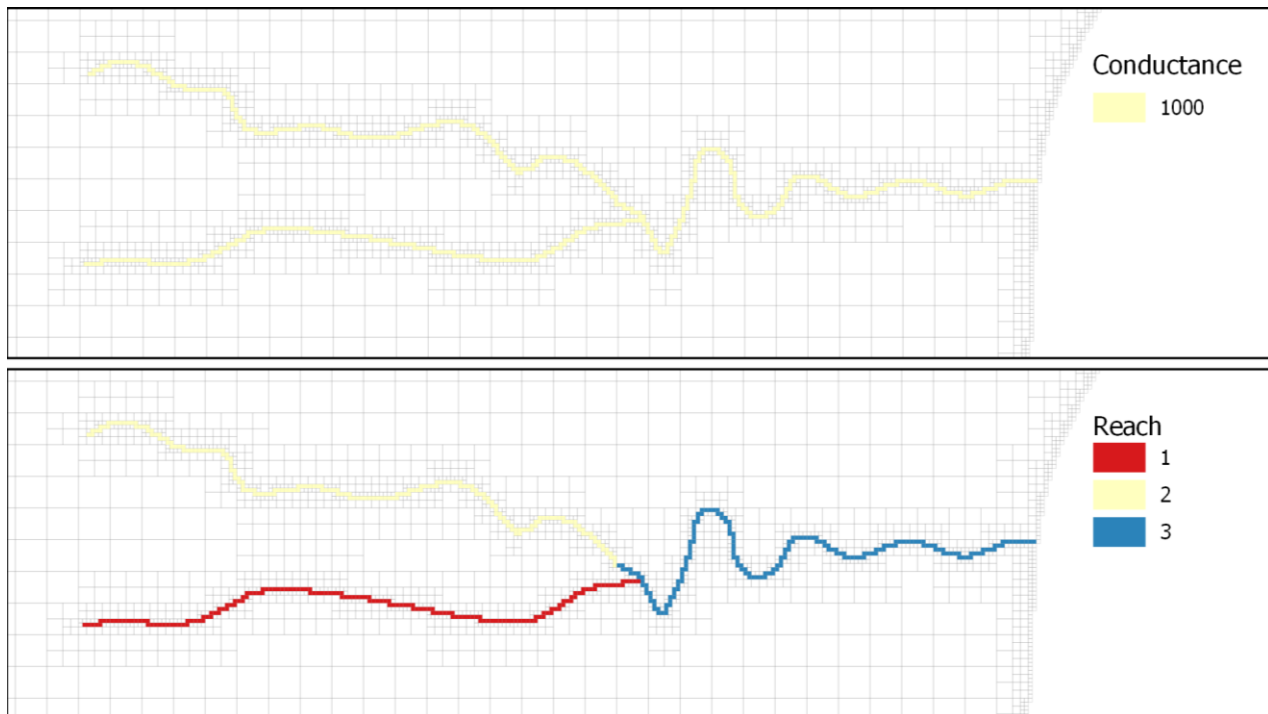


Figure 2 Plan view of the model grid DRN cells coloured according to conductance (top) and stream reach number (bottom).

3. INTERPOLATING TO LINEAR FEATURES

This tutorial demonstrates several ways to interpolate from pilot points to DRN cells that represent a polylinear feature such as a stream. In the present case, some of these streams are relatively straight while others are a bit bendy.

The tutorial starts with “simple” pilot point interpolation without zonation. Then zonation is included to distinguish between stream reaches. Lastly, SEGLISTS are introduced. These allow a more complex methodology to be employed for pilot point to feature interpolation; this methodology is better for handling bendy streams.

3.1 The Easy Way (Using Pilot Points)

We will start with “simple” pilot point to stream interpolation. In this case we will simply apply inverse power of distance interpolation from pilot points to all reaches of a stream. For this, we need a set of pilot points. We must also set up a PLPROC input file.

The Pilot Points

Pilot points would normally be digitised (either manually or automatically) to create a “list file” – a type of file that is easily read by PLPROC. (See other GMDSI tutorials for descriptions of how to build this types of file.)

For the sake of expediency, pilot point identifiers, coordinates, and associated hydraulic property values are provided in file *drainpp.dat*. Open this file using a text editor and take a look. You should see four columns. Each row pertains to an individual pilot point; each pilot point has a unique identifier (*ppid*), coordinates (*easting* and *northing*) and a value of drain conductance (*draincond*).

Where PEST is used for model calibration, a template of such a file should be constructed so that PEST can transfer current parameter values to each pilot point prior to running the model. Under

these circumstances, PLPROC is run using a command issued in a batch file which PEST runs as “the model”.

Figure 3 displays pilot points featured in *drainpp.dat* together with the value for drain conductance assigned to each of them; part of the model grid is also shown. . Pilot point locations used here were selected manually. Their placement was roughly in accordance with the following principles:

- (a) Employ sufficient pilot points to follow the principles of highly parameterized inversion – i.e., we do not want parameter parsimony.
- (b) Place pilot points close to the stream (although it does not need to be very precise).
- (c) Ensure a higher density of pilot points where stream branches join. This prevents the conductance in one branch from being influenced too much by the conductance of a pilot point associated with another branch. (This is the main problem with the “simple” pilot point approach).

For convenience, the conductance value for each pilot point was arbitrarily assigned an increasing value according to position in the list (from 1 to 57).

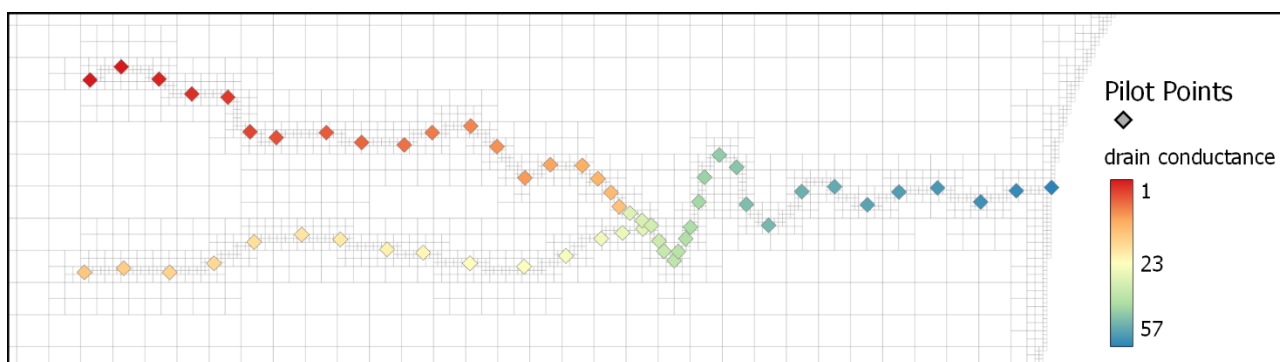


Figure 3 Model grid and locations of manually emplaced pilot points. Note the higher density of pilot points where the three branches of the stream join.

The PLPROC Input File

This section demonstrates how to construct a PLPROC input file to interpolate from pilot points to the model drain cells. We assume that you are at least somewhat familiar with PLPROC and its function; nevertheless, while reading this section, you may benefit from having the PLPROC manual close at hand. Other [GMDSI tutorials](#) go into greater detail on how to set up and use PLPROC scripts.

We must write a script which accomplishes the following five 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 input files.

Let’s get started.

- 5 Open an empty text file. Name it *plproc_drains1.dat* and type in the function below to read model grid specifications:

```
Cl_mf6 = read_mf6_grid_specs(file=model.disv.grb,      &
                             dimensions=3,             &
                             slist_layernum = layer,   &
                             slist_idomain  = idomain)
```

- 6 This function tells PLPROC to create a CLIST (i.e a coordinate list) named *cl_mf6*, based on the contents of the binary grid file *model.disv.grb* written by MODFLOW 6. Even though the MODFLOW 6 model is of the DISV type, we have chosen to represent it (internally to PLPROC) as a three-dimensional rather than two-dimensional grid. This function also creates ancillary SLISTS for cell IDOMAIN values and LAYER numbers (although the latter are not relevant for the current exercise). An SLIST (for “selection list”) is a list of integer values that are commonly used for identification of subgroups of an associated PLIST).
- 7 Next, we need to create an SLIST that identifies cells to which DRN boundary conditions have been assigned. This SLIST is named *draincells_mf6*. This can be accomplished this using the *find_cells_in_lists()* function. This function reads the MODFLOW 6 input file containing DRN data (*model.drn_stress_period_data_1.txt*). Add the following function to the input file. (Cut and paste from this document if that helps).

```
draincells_mf6 = cl_mf6.find_cells_in_lists(           &
    file=model.drn_stress_period_data_1.txt,         &
    model_type=mf6_disv,                             &
    list_col_start=1,                                 &
    keytext_start='top_of_file',                      &
    keytext_end='end_of_file')
```

- 8 Because the *model_type* is designated as “mf6_disv”, PLPROC knows that cells in this file are identified by layer number and layer index (i.e. *icpl*). Because *list_col_start* is set to 1, it knows that layer numbers are read from column 1 and that cell *icpl* values are read from column 2. Armed with this information, PLPROC reads all rows of file *model.drn_stress_period_data_1.txt* to create a list of DRN cells. These are stored in an SLIST named *draincells_mf6* for further use.
- 9 Now that we have read the model geometry and identified drain cells, we need to obtain the locations of pilot points, and the conductance values that are associated with them. We can accomplish this using the *read_list_file()* function. Add the following to the PLPROC input file:

```
cl_pp = read_list_file(file='drainpp.dat',           &
    id_type='integer',                               &
    skiplines=1,                                     &
    dimensions=2,                                    &
    plist='draincond_pp';column=4)
```

- 10 Using data contained in the list file *drainpp.dat*, we have created a new CLIST for pilot points named *cl_pp*. We have also created an associated PLIST (a PLIST is a “parameter list”) using conductance values featured in the fourth column of the list file.
- 11 Next, we must create a PLIST associated with the model grid CLIST (*cl_mf6*). This is the PLIST to which drain cell conductances will be interpolated from pilot points; however, as we will see shortly, interpolation will only take place to cells in which a DRN boundary condition has been emplaced in the MODFLOW 6 model. Add the following function to the PLPROC input file:

```
draincond_mf6=new_plist(reference_clist=cl_mf6,value=0.0)
```

- 12 Now we can interpolate conductance values from the pilot points PLIST to the model cells PLIST. For simplicity, we shall employ inverse power of distance interpolation ; however, any interpolation method could be used. Through use of the “selection equation” on the left side of this function,

interpolation takes place only to those model cells to which a DRN boundary condition has been assigned. Add the following function to the PLPROC input file:

```
draincond_mf6(select=(draincells_mf6.ne.0))=draincond_pp.ivd_interpolate_2d(&
    transform='log',                                &
    inv_power=2.0,                                  &
    min_points=2,                                    &
    max_points=20,                                   &
    search_radius=1.0e20)
```

- 13 Finally, we must write a new model input file containing model conductance values interpolated from pilot points, these being contained in the *draincond_mf6* PLIST. MODFLOW 6 will be directed to read this new file when running a simulation. In our case, we use the *replace_cells_in_lists()* function (see the PLPROC manual for further details on this function). Add the following function to the PLPROC input file:

```
replace_cells_in_lists(old_file=model.drn_stress_period_data_1.txt,    &
    new_file=new_model.drn_stress_period_data_1.txt,                  &
    model_type=mf6_disv,                                              &
    list_col_start=1,                                                  &
    keytext_start='top_of_file',                                       &
    keytext_end='bottom_of_file',                                      &
    plist=draincond_mf6;column=4;action='replace')
```

- 14 This function instructs PLPROC to write a new file named *new_model.drn_stress_period_data_1.txt*, based on the original model DRN package file (*model.drn_stress_period_data_1.txt*). Conductance values in the fourth column of the original file are replaced by interpolated conductance values contained in the *draincond_mf6* PLIST.
- 15 The argument *model_type=mf6_disv* used in the above function instructs PLPROC that the DRN package input file pertains to a MODFLOW 6 model with a DISV grid. Hence, as stated above, cells are defined by layer number and cell index (in that order). The *list_col_start=1* argument informs PLPROC that layer numbers reside in the first column of this file and (therefore implicitly) that cell indices reside in the second column. As the list comprises the only contents of the file, the *keytext_start* and *keytext_end* arguments are set to 'top_of_file' and 'bottom_of_file' respectively. Finally, the *plist* argument specifies that the fourth column (4) must be modified (*replace*) by values in the PLIST *draincond_mf6*.
- 16 Lastly, just for our own interest (and to check that we have done everything correctly), let us add a reporting function to the PLPROC script. Add the following to the PLPROC input file:

```
cl_mf6.report_dependent_lists(file='report.dat')
```

- 17 Great, we are now ready to run PLPROC for the first time. Make sure to save all changes in file *plproc_drains1.dat*.
- 18 Open a command prompt in your working folder. Execute PLPROC by typing: *plproc plproc_drains1.dat* and pressing <enter>. You should see the following:

Reading and storing contents of PLPROC script file plproc_drains1.dat...
Processing commands in PLPROC script file...

```
> cl_mf6=read_mf6_grid_specs(file=model.disv.grb,dimensions=3,slist_lay...
> draincells_mf6=cl_mf6.find_cells_in_lists(file=model.drn_stress_perio...
> cl_pp=read_list_file(file='drainpp.dat',id_type='integer',skiplines=1...
> draincond_mf6=new_plist(reference_clist=cl_mf6,value=0.0)
> draincond_mf6(select=(draincells_mf6.ne.0))=draincond_pp.ivd_interpol...
> replace_cells_in_lists(old_file=model.drn_stress_period_data_1.txt,ne...
> cl_mf6.report_dependent_lists(file='report.dat')
```

End of file: no more commands to process.

- 19 Check your working folder. You should have two new files: *new_model.drn_stress_period_data_1.txt* and *report.dat*. Open them both in a text editor and take a look.
- 20 Now compare *new_model.drn_stress_period_data_1.txt* to the original DRN package input file *model.drn_stress_period_data_1.txt*. Note how the structure is the same, however the values in the fourth column of the new file have been altered by PLPROC.
- 21 Display these new DRN conductance values on a map to see the outcomes of the interpolation process. Repeat the steps described in chapter **2.2 Viewing the Model**. (Note that the provided batch file will not work for the new files because of their new names). You should be able to display something like this (Figure 4):

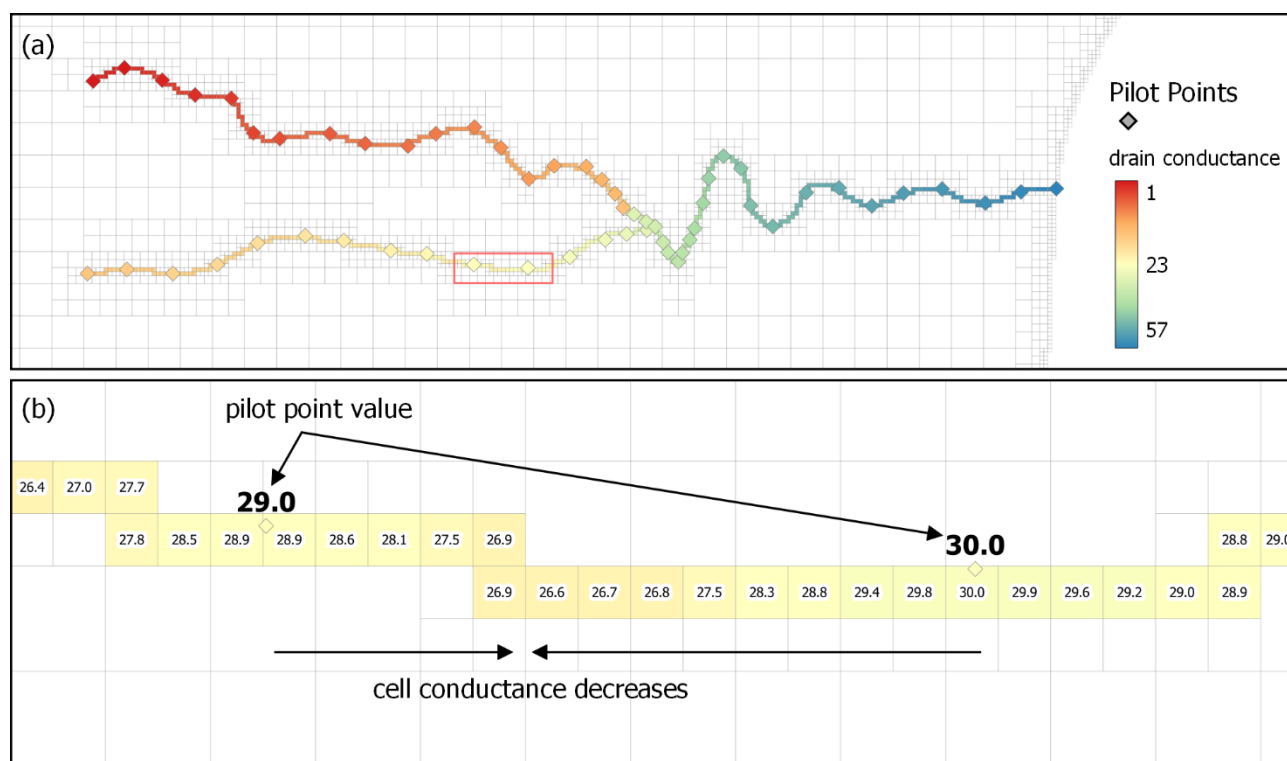


Figure 4 (a) Model drain cells with spatially varying conductance values interpolated from pilot points; and (b) expanded view of a section of reach 1 (labels designate pilot point and cell conductance values) to highlight issues with this method.

As you can see in Figure 4a, interpolated conductances of model drain cells appear to match pilot point conductances. However, a closer look at reach 1 (see Figure 4b) highlights an issue with this approach. Figure 4b shows a close-up look of a section of reach 1, between two pilot points that have

conductance values of 29.0 and 30.0. We expect drain conductances to increase monotonically from one pilot point all the way to the other (i.e., to increase steadily from 29.0 to 30.0). However, what actually occurs is that drain conductance decreases with distance from both pilot points, reaching a low-point midway between the two.

This discrepancy is caused by the influence of pilot points near reach 2 (which have lower conductance values). As an infinite search radius was used during interpolation to avoid discontinuities as points move in and out of the search distance, interpolated drain cell conductances can be influenced by pilot points that are far away. The influence is small; in many cases this will not matter too much. The following sections demonstrate alternative interpolation methods that avoid this problem at the expense of a slightly more complicated setup.

All files generated so far can be found in the folder *t2*. This folder also contains batch files (**.bat*) that automate the model display steps.

3.2 The Slightly Harder Way (Using Pilot Points and Zones)

To preclude pilot points in one reach from influencing cell conductances in other reaches, we can associate pilot points with reach-specific groups of cells. One way to achieve this is by providing a pilot point input file for each reach; at the same time, PLPROC must be informed of which cells belong to which reach.

In the tutorial folder three pilot point files have been provided for you. These are named *drainpp_1.dat*, *drainpp_2.dat* and *drainpp_3.dat*. The filename contains the respective stream reach number (e.g., *drainpp_1.dat* refers to reach 1). If you compare these files to the pilot point file used in the previous chapter (*drainpp.dat*), you will see that pilot points have been divided amongst the new files according to their proximity to each reach.

Additionally, where reach 1 and 2 meet reach 3, a pilot point from reach 3 is repeated in the pilot point files pertaining to the other reaches. As you can see from Figure 5, this means that *drainpp_1.dat* contains pilot point 37 and *drainpp_2.dat* contains pilot point 35. Pilot points 35 and 37 are also included in file *drainpp_3.dat*. The conductance value associated with these two pilot points does not have to be the same in each of the files. However, if the same value is used in each file, then interpolated model conductances will not be discontinuous where the reaches meet (as happens for reach 2 and 3 in Figure 4a). Another option is to associate different pilot point parameters with the same point in each of the files, but tie the parameters tied to each other in the PEST control file. This might not even be a concern. It depends on the case.

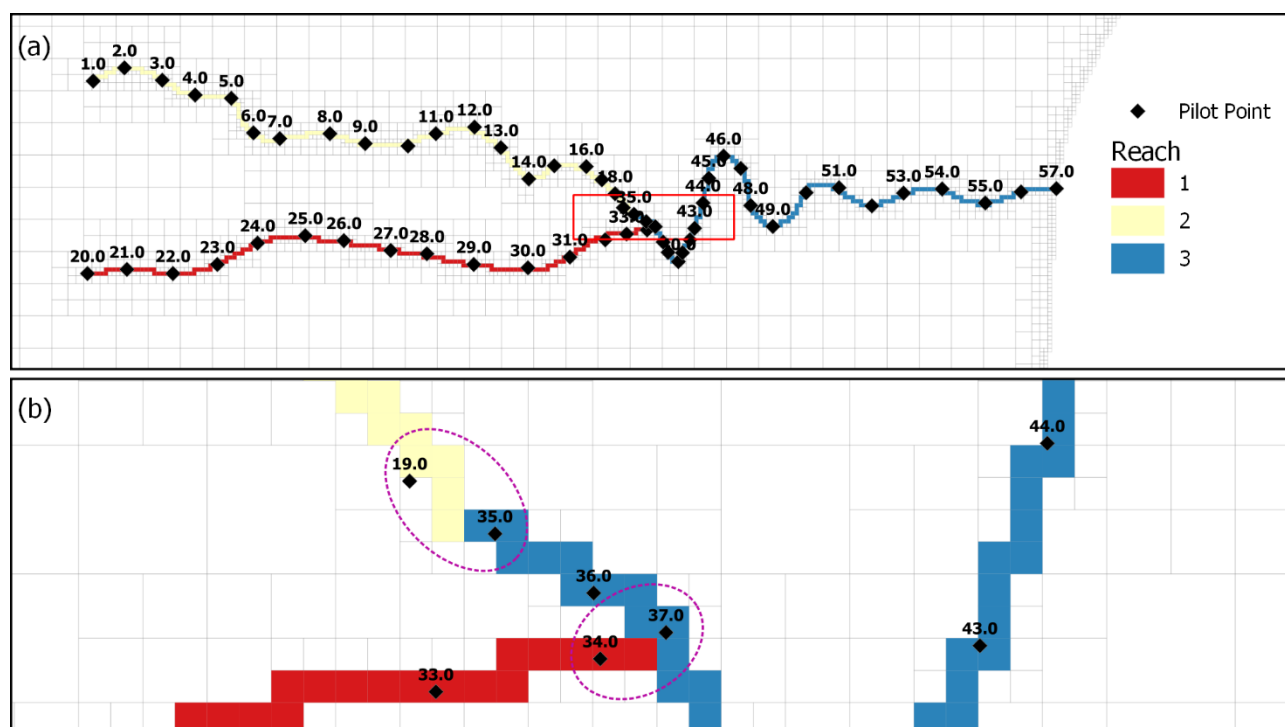


Figure 5 (a) Stream reaches and pilot points; pilot points are labelled according to *ppid* number. (b) Close-up look at the confluence of the reaches. Purple dotted circles highlight pilot points that are repeated in the respective pilot point list files.

Now we need to set up a PLPROC input file that accounts for reach-specific interpolation. This file will follow the same principles as the input file created in the previous chapter. Let us get started.

22 Create a new text file. Name it *plproc_drains2.dat*.

23 Repeat step 5 to create the model cell CLIST named *cl_mf6*.

- 24 Unfortunately, the original model file was set up in a way that cell stream reach numbers are identified using a text string (e.g., *stream-1*, *stream-2*, etc.) in column 5 (see *model.drn_stress_period_data.txt*). While these text strings were used by the modeller purely as a convenience (it is easy to understand what they refer to), he/she could just as easily have used an integer. However, PLPROC requires that this column contain only integers or real numbers, so first we need to edit it. (Keep this in mind when setting up a MODFLOW model in a graphical user interface of FloPy as it will reduce the need for later manual file alterations.)
- 25 Make a copy of the original file *model.drn_stress_period_data.txt*. Name the copy *drains.txt*. Open *drains.txt* in a text editor and remove the sub-string “*stream-*” from the 5th column. The first few rows of the modified *drains.txt* file should now look like this:

1	2830	140.94892883	1000.00000000	1
1	2842	139.41760254	1000.00000000	1
1	2843	138.98307800	1000.00000000	1
1	2845	139.76692200	1000.00000000	1

- 26 Save *drains.txt* in your working folder and close it.
- 27 Now return to the input file *plproc_drains2.dat*.
- 28 Next create the SLIST for the drain cells (similar to step 7) using the *find_cells_in_list()* function and the *drains.txt* list file. This time we need to populate the SLIST with cell reach numbers so that we can distinguish between cells in different reaches. Type in the following function:

```
draincells_mf6 = cl_mf6.find_cells_in_lists(                                &
    file=drains.txt,                                                       &
    model_type=mf6_disv,                                                  &
    list_col_start=1,                                                      &
    assign_col = 5,                                                        &
    keytext_start='top_of_file',                                          &
    keytext_end='bottom_of_file')
```

- 29 As you can see, the major difference between use of the *find_cells_in_list()* function as used in the present chapter and in the preceding chapter is the inclusion of the argument *assign_col = 5*. If this argument is omitted, then model cells are assigned a value of 1 if they appear in the list, and 0 if they do not. However the *assign_col = 5* argument instructs PLPROC to assign cells appearing in the list values that are read from column 5 of the list; these values are assigned to the *draincells_mf6* SLIST.
- 30 Now create three pilot point CLISTs; one for each pilot point file. Start with *drainpp_1.dat*. Type in the following (or cut and paste from this document):

```
cl_pp1 = read_list_file(file='drainpp_1.dat',                             &
    id_type='integer',                                                    &
    skiplines=1,                                                          &
    dimensions=2,                                                         &
    plist='draincond1_pp';column=4)
```

- 31 As you can see, this function call follows the same protocol as was used in the previous chapter (see step 9). The *read_list_file()* function is used to read the pilot point file *drainpp_1.dat* to create a CLIST (in this case *cl_pp1*) and an associated PLIST (*draincon1_pp*).

32 Do the same for the remaining two pilot point files (*drainpp_2.dat* and *drainpp_3.dat*). Name the CLISTs *cl_pp2* and *cl_pp3* and the respective PLISTs *draincond2_pp* and *draincond3_pp*. The function calls are as follows:

```
cl_pp2 = read_list_file(file='drainpp_2.dat',      &
                        id_type='integer',        &
                        skiplines=1,              &
                        dimensions=2,             &
                        plist='draincond2_pp';column=4)

cl_pp3 = read_list_file(file='drainpp_3.dat',      &
                        id_type='integer',        &
                        skiplines=1,              &
                        dimensions=2,             &
                        plist='draincond3_pp';column=4)
```

33 Next initialize the model drain conductance PLIST (see step 11).

```
draincond_mf6=new_plist(reference_clist=cl_mf6,value=0.0)
```

34 Now we can interpolate conductance values from each of the three pilot point PLISTs to the model cells PLIST. This step differs from that provided in the previous chapter in that interpolation is reach-specific. Add the following lines to the PLPROC input file:

```
draincond_mf6(select=(draincells_mf6.eq.1))=      &
                                                    draincond1_pp.ivd_interpolate_2d( &
                                                    transform='log',          &
                                                    inv_power=2.0,              &
                                                    min_points=2,                &
                                                    max_points=20,                &
                                                    search_radius=1.0e20)
```

35 As you can see, this command is very similar to that used in the previous chapter. The main difference is the design of the selection equation (*select=(draincells_mf6.eq.1)*). This specifies that interpolation take place only to model cells for which elements in the *draincells_mf6* SLIST and have a value equal (.eq) to 1. Recall that this is the reach number.

36 Do the same for reaches 2 and 3 using the respective selection equations. Function calls are as follows:

```
draincond_mf6(select=(draincells_mf6.eq.2))=      &
                                                    draincond2_pp.ivd_interpolate_2d( &
                                                    transform='log',          &
                                                    inv_power=2.0,              &
                                                    min_points=2,                &
                                                    max_points=20,                &
                                                    search_radius=1.0e20)

draincond_mf6(select=(draincells_mf6.eq.3))=      &
                                                    draincond3_pp.ivd_interpolate_2d( &
                                                    transform='log',          &
                                                    inv_power=2.0,              &
                                                    min_points=2,                &
                                                    max_points=20,                &
                                                    search_radius=1.0e20)
```

- 37 Finally, repeat step 13 to write the PLIST of model drain conductances to a new model input file. Name the new file *new2_model.drn_stress_period_data_1.txt*.

```
replace_cells_in_lists(old_file=model.drn_stress_period_data_1.txt,      &
                      new_file=new2_model.drn_stress_period_data_1.txt,  &
                      model_type=mf6_disv,                             &
                      list_col_start=1,                                &
                      keytext_start='top_of_file',                     &
                      keytext_end='bottom_of_file',                    &
                      plist=draincond_mf6;column=4;action='replace')
```

- 38 Great, we are now ready to run PLPROC for the second time. Make sure to save all changes in the *plproc_drains2.dat* PLPROC input file.

- 39 Open a command prompt in your working folder. Execute PLPROC by typing: *plproc plproc_drains2.dat* and pressing <enter>. You should see the following:

```
> cl_mf6=read_mf6_grid_specs(file=model.disv.grb,dimensions=3,slist_lay...
> draincells_mf6=cl_mf6.find_cells_in_lists(file=drains.txt,model_type=...
> cl_pp1=read_list_file(file='drainpp_1.dat',id_type='integer',skipline...
> cl_pp2=read_list_file(file='drainpp_2.dat',id_type='integer',skipline...
> cl_pp3=read_list_file(file='drainpp_3.dat',id_type='integer',skipline...
> draincond_mf6=new_plist(reference_clist=cl_mf6,value=0.0)
> draincond_mf6(select=(draincells_mf6.eq.1))=draincond1_pp.ivd_interpo...
> draincond_mf6(select=(draincells_mf6.eq.2))=draincond2_pp.ivd_interpo...
> draincond_mf6(select=(draincells_mf6.eq.3))=draincond3_pp.ivd_interpo...
> replace_cells_in_lists(old_file=model.drn_stress_period_data_1.txt,ne...
> cl_mf6.report_dependent_lists(file='report.dat')
```

End of file: no more commands to process.

- 40 Check your working folder. You should have two new files: *new2_model.drn_stress_period_data_1.txt* and *report.dat*. Open them both in a text editor and take a look.

- 41 Display these values on a map to see interpolation outcomes. Repeat the steps described in chapter **2.2 Viewing the Model** to display drain conductance. (Note that the provided batch file will not work for the new files because of altered file names). You should be able to display something similar to Figure 6.

Compare Figure 6b to Figure 4b; note how in Figure 6b model-based cell drain conductance values change monotonically between pilot points. However, if you take a closer look at cell conductances where the stream is bendy (see Figure 7) you will see that this method does not always respect the stream profile. This may not be an issue in some modelling contexts. Nevertheless, it can be avoided using the methodology described in the following chapter.

All files generated up to this point are in the *t3* folder. The folder also contains batch files (**.bat*) to automate the model display steps.

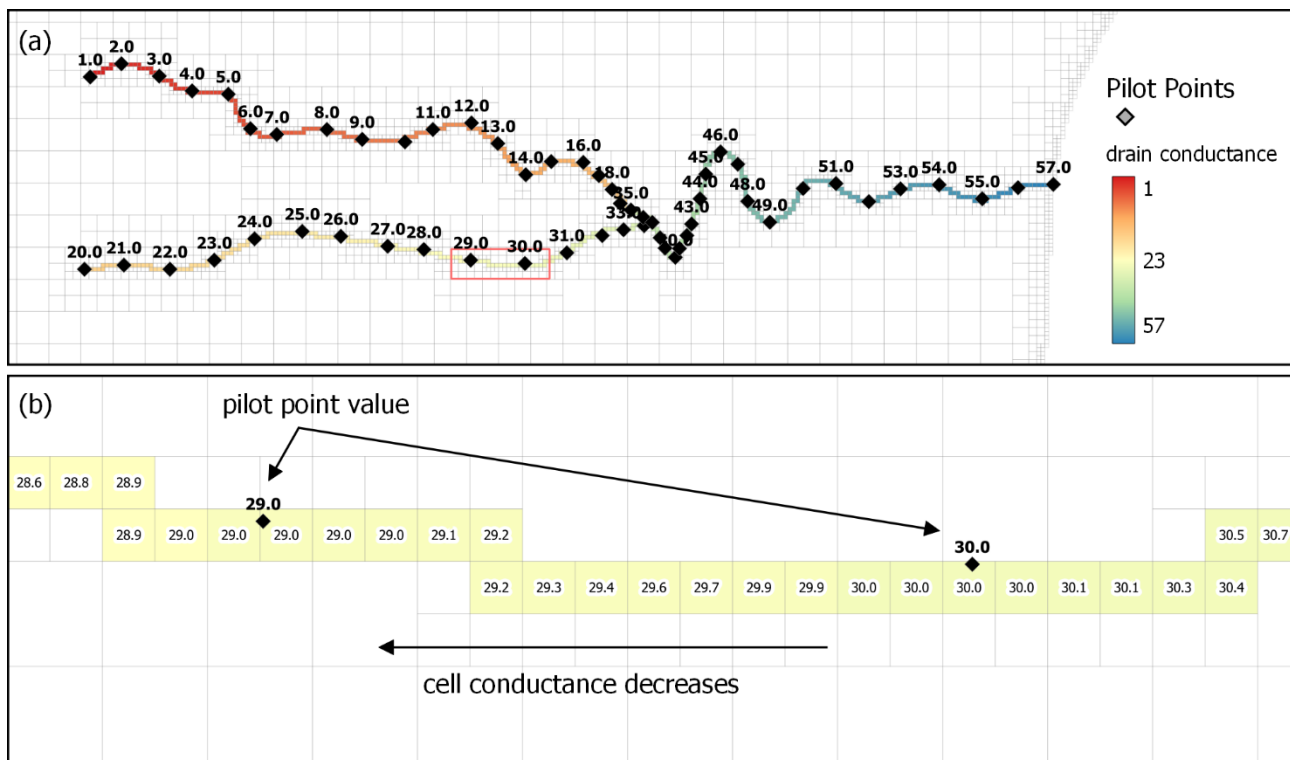


Figure 6 (a) Model drain cells with spatially varying conductance values interpolated from reach-specific pilot points; and (b) expanded view of a section of reach 1 (labels designate pilot point and cell conductance values).

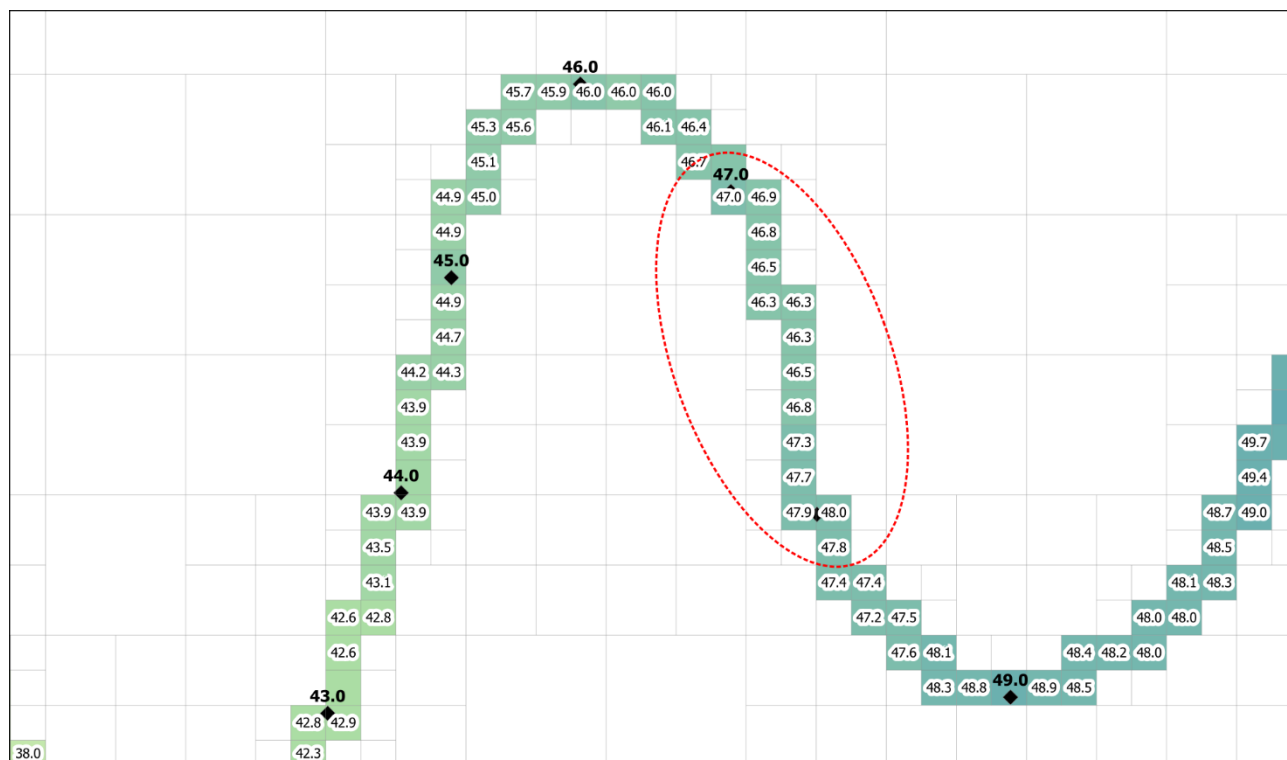


Figure 7 Close-up view of model drain cells with spatially varying conductance values interpolated from reach-specific pilot points; (labels designate pilot point and cell conductance values).

3.3 Using SEGLISTS to Respect Stream Geometry

Suppose that we have really bendy streams (similar to reach 3) and that we want to do linear interpolation along the profile of these streams between pilot points. (By “linear” we mean with respect to distance along the stream, regardless of its curviness) with interpolation taking place between only two pilot points at a time.) To accomplish this, links between pilot points and stream segments, as well as the ordering of pilot points and segments, must be registered with PLPROC. So, for every model drain cell we need to know the two pilot points from which interpolation takes place. This is where SEGLISTS come in.

The use of SEGLISTS requires a slightly different approach than the more generic use of pilot points described in previous chapters. This workflow requires two new steps: (1) defining the geometry of the SEGLISTS and (2) generating appropriately-placed pilot points from the SEGLISTS.

PLPROC needs to be informed of the geometry of the polylinear features (e.g., the streams) to which interpolation from pilot points must take place, as well as how to break these features up into segments. This is accomplished with a SEGLIST (which will be described further on). The size and spacing of segments comprising a SEGLIST should reflect the propensity for heterogeneity in model parameters (in this case DRN conductances). Segment design also needs to account for connections and connections between different linear features (for example, the confluence of two streams).

Once a SEGLIST has been created from a group of juxtaposed segments, PLPROC can generate a family of pilot points located at segment joins and at either end of the group of joined segments. PLPROC is clever enough to arrange things so that the same pilot point can serve more than one segment where they join, this making setup more efficient. After building the SEGLIST and associated pilot points, interpolation to model cells is accomplished in a similar fashion to that described in previous chapters.

Let us get started.

Building the SEGLIST

A SEGLIST is a set of segments. Each segment is a polyline defined by a set of vertices (e.g., a set of coordinates). A segment begins at the first vertex of the vertex list that is attributed to a segment and ends at the last vertex in the list. Each segment within a SEGLIST has a name (of 20 characters or less in length).

Vertices for the streams which feature in our model have been provided in the tutorial folder in a file named *seglists.csv*. These were obtained simply by defining stream polyline vertices in QGIS and by saving them as a CSV file. They could just as easily have been digitized manually or generated at specified intervals along the stream polyline. The spatial density of these vertices should be greater than the desired spatial density of pilot points. Furthermore, it is convenient to place a vertex where linear features join or intersect.

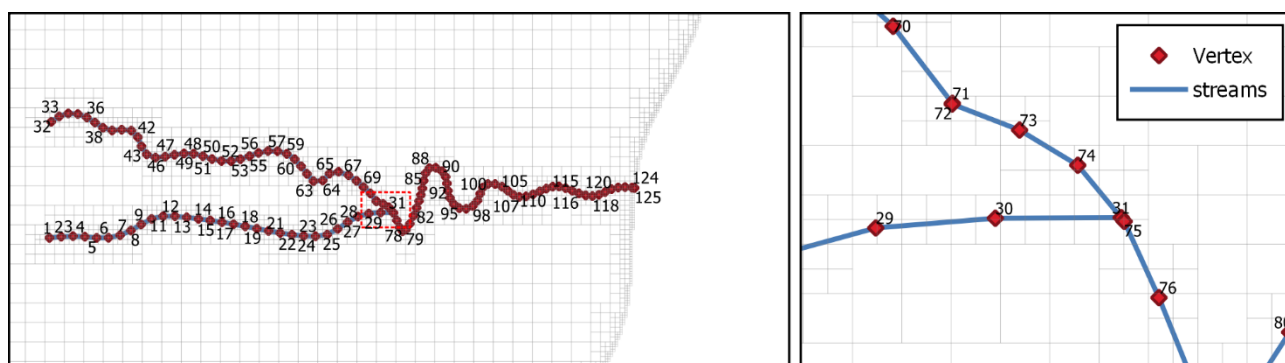


Figure 8 Stream polyline and vertices used to build SEGLISTS. Vertices are labelled by index number.

PLPROC reads SEGLIST vertex coordinates and names from a segment file, called a “SEGFILE” for short. Now that we have all the vertex coordinates, they need to be grouped into segments in a SEGFILE. How to group the vertices into segments is up to the modeller. He/she should keep in mind that a pilot point will exist at the beginning and end of each segment. It is also convenient for segment endpoints to coincide with points where polylinear features such as streams join or intersect.

An example SEGFILE (named *segfile.dat*) is provided in the tutorial folder. A segment file can adopt either of two protocols, namely “block” or “table”. *segfile.dat* employs the “table” protocol as it is easier to manipulate and display data contained in this file in GIS software. See the PLPROC manual for further details. Open *segfile.dat* in a text editor or spreadsheet package and take a look.

Note that each segment within a SEGLIST is identified by a unique name in the *segid* column. Segment vertices are ordered sequentially. Where segments join, the last vertex of the one segment is repeated as the first vertex of the adjoining segment. Figure 9 displays stream vertices colour coded according to their assigned segment. As you can see in Figure 9b, where three segments join, the vertex at which the join occurs is repeated in each of the adjoining segments.

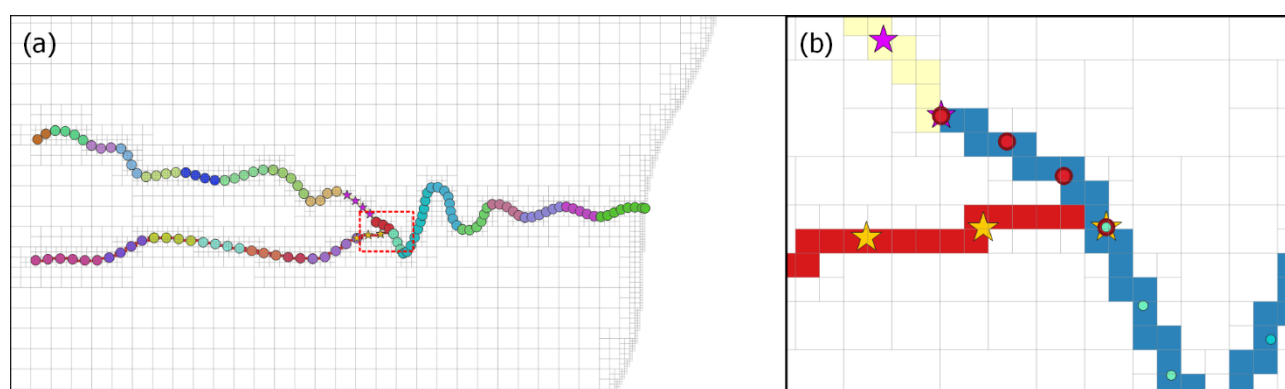


Figure 9 (a) All stream vertices grouped according to segment, and (b) a close-up look at the intersection of three segments. (Model drain cells are coloured according to stream reach).

Now that we have a SEGFILE that contains SEGLISTS, we can use PLPROC to generate a family of pilot points located at segment joins. We could also generate these pilot points ourselves if we really wanted to – but because PLPROC can do the hard work, let us make use of it. To get started, we need to build a PLPROC input file.

42 Open a new text file in your text editor of choice. Name the new file *plproc_drains3.dat*.

43 As in previous chapters, start by creating a model cell CLIST. Use the following function.

```
cl_mf6 = read_mf6_grid_specs(file=model.disv.grb,      &
                             dimensions=3,            &
                             slist_layernum = layer,  &
                             slist_idomain  = idomain)
```

44 Next, create a SEGLIST named *sl_drains*, by reading the provided SEGFILE (*segfile.dat*) using PLPROC’s *read_segfile()* function. Insert the following function into the PLPROC script.

```
sl_drains = read_segfile(file="segfile.dat")
```

45 Next, instruct PLPROC to create a CLIST for pilot points, using the *sl_drains* SEGLIST as its base. Nodes of this CLIST will be created at the ends of each segment of the SEGLIST. This is

accomplished using the *create_clist_from_seglist()* function. Type in the function as follows (or copy and paste from this document.)

```
cl_drainpp = create_clist_from_seglist(seglist=sl_drains,      &
                                     linkage_type=endpoints,  &
                                     dist_thresh=10.0)
```

- 46 This function creates a CLIST named *cl_drainpp*, based on the SEGLIST named *sl_drains*. The argument *linkage_type* specifies the nature of SEGLIST-to-CLIST linkage. Linkages can be *endpoints* or *midpoint*. If linear interpolation along each segment is desired (as in our case), then *endpoints* is preferred option. If a single piecewise constant value per segment is desired, then the *midpoint* option should be used. See the PLPROC manual for more details.
- 47 If *endpoints* are used as the *linkage_type*, then a value for the *dist_thresh* argument must be provided. If the ends of two or more segments are separated by less than *dist_thresh*, only a single CLIST element (i.e. a pilot point in this case) is created; this is linked to all SEGLIST segments which terminate at, or close to, that point. In our case, if the endpoints of two segments are less than 10 m apart, the same pilot point is used for both segments. This allows properties interpolated from pilot-point-interpolated parameters to vary continuously between connected segments.
- 48 Optionally, we can access details on the newly-created set of pilot points by instructing PLPROC to report on the *cl_drainpp* CLIST. Do this by inserting the following function into the PLPROC script.

```
cl_drainpp.report_dependent_lists(file='report.dat')
```

We can now build a pilot point list file by running PLPROC and editing the contents of *report.dat*.

- 49 Save *plproc_drains3.dat*.
- 50 Open the command line in your working folder, type *plproc plproc_drains3.dat* and press <enter>. If all went well, you should see something like the following.

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

> cl_mf6=read_mf6_grid_specs(file=model.disv.grb,dimensions=3,slst_lay...
> sl_drains=read_segfile(file="segfile.dat")
> cl_drainpp=create_clist_from_seglist(seglist=sl_drains,linkage_type=e...
> cl_drainpp.report_dependent_lists(file='report.dat')

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

Generating the Pilot Points

File *report.dat* should now exist in your working folder. Open it in a text editor and take a look. You should see four columns, the last two of which are X and Y coordinates. These are the locations of CLIST elements (pilot points in our case). Now we can create a pilot point list file by simply copying from *report.dat*. We don't need to record pilot point X and Y coordinates in this file as we did in previous pilot point list files. We will ask PLPROC to read the contents of this file into an existing CLIST (i.e. the CLIST created as above).

- 51 Open a new blank file in either a text editor or spreadsheet package. Name it *draincond.dat*.
- 52 Copy the values in the first column of *report.dat* (the column named “*Index*”) into file *draincond.dat*. Name this column “*ID*”.
- 53 Create a second column named “*drain_conductance*”. Fill this column with whatever values you like. Once you are done, the first few rows of your file should look something like this:

```
ID  drain_conductance
1   10
2   20
3   40
4   55
```

- 54 The file *draincond.dat* is a list file which contains pilot point identifiers in the first column (*ID*) and conductance values in the second column (*drain_conductance*).

At this stage, all of the files needed for the final PLPROC script are ready. The following steps for setting up the PLPROC input file are similar to those provided in the previous chapters.

- 55 Open *plproc_drains3.dat* in your text editor again.
- 56 Instruct PLPROC to read the *draincond.dat* list file to obtain conductance values at pilot points by inserting the following function into the PLPROC script.

```
read_list_file(reference_clist='cl_drainpp',skiplines=1,      &
               file='draincond.dat',                      &
               plist='pp_draincond';column=2)
```

- 57 Next, instruct PLPROC to build an SLIST of model drain cells to which interpolation must take place (similar to step 11).

```
draincells_mf6 = cl_mf6.find_cells_in_lists(                &
               file=model.drn_stress_period_data_1.txt,    &
               model_type=mf6_disv,                        &
               list_col_start=1,                            &
               keytext_start='top_of_file',                 &
               keytext_end='end_of_file')
```

- 58 Now calculate interpolation factors to these model cells through linear interpolation along the segments. (This function needs to be run only once because linear interpolation factors are independent of the values of quantities which they interpolate. In practice, when incorporated into a PEST/PEST++ workflow, it would usually be run once and then commented out.)

```
calc_linear_interp_factors(source_clist=cl_drainpp,        &
                           target_clist=cl_mf6;select=(draincells_mf6.ne.0), &
                           file="factors_draincells.dat", &
                           search_radius=100)
```

- 59 Initialize the drain conductance PLIST.

```
draincond_mf6=new_plist(reference_clist=cl_mf6,value=0.0)
```

60 Undertake interpolation using the *interp_using_file()* function.

```
draincond_mf6=pp_draincond.interp_using_file(file=factors_draincells.dat, &
                                             transform=log)
```

61 And finally, write the values in to a model input file.

```
replace_cells_in_lists(old_file=model.drn_stress_period_data_1.txt,      &
                       new_file=new3_model.drn_stress_period_data_1.txt,  &
                       model_type=mf6_disv,                             &
                       list_col_start=1,                                 &
                       keytext_start='top_of_file',                     &
                       keytext_end='bottom_of_file',                     &
                       plist=draincond_mf6;column=4;action='replace')
```

Great. We are now ready to run PLPROC for the last time. Make sure to save all changes in the *plproc_drains3.dat* PLPROC input file.

62 Open a command prompt in your working folder. Execute PLPROC by typing: *plproc plproc_drains3.dat* and pressing <enter>. You should see the following:

```
Reading and storing contents of PLPROC script file plproc_drains3.dat...
Processing commands in PLPROC script file...

> cl_mf6=read_mf6_grid_specs(file=model.disv.grb,dimensions=3,slist_lay...
> sl_drains=read_segfile(file="segfile.dat")
> cl_drainpp=create_clist_from_seglist(seglist=sl_drains,linkage_type=e...
> cl_drainpp.report_dependent_lists(file='report.dat')
> read_list_file(reference_clist='cl_drainpp',skiplines=1,file='drainco...
> draincells_mf6=cl_mf6.find_cells_in_lists(file=model.drn_stress_perio...
> calc_linear_interp_factors(source_clist=cl_drainpp,target_clist=cl_mf...
> draincond_mf6=new_plist(reference_clist=cl_mf6,value=0.0)
> draincond_mf6=pp_draincond.interp_using_file(file=factors_draincells....
> replace_cells_in_lists(old_file=model.drn_stress_period_data_1.txt,ne...

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

63 Check your working folder. You should have two new files: *new3_model.drn_stress_period_data_1.txt* and *factors_draincells.dat*. Open them both in a text editor and take a look.

64 Display the new model drain cell conductance values on a map to see the outcome of the interpolation. Repeat the steps described in chapter **2.2 Viewing the Model** to display drain conductance. (Note that the provided batch file will not work for the new files). You should be able to display something similar to Figure 10 (your drain conductance values will depend on what conductance values you assigned to pilot points in step 53; they will probably be different from those displayed below).

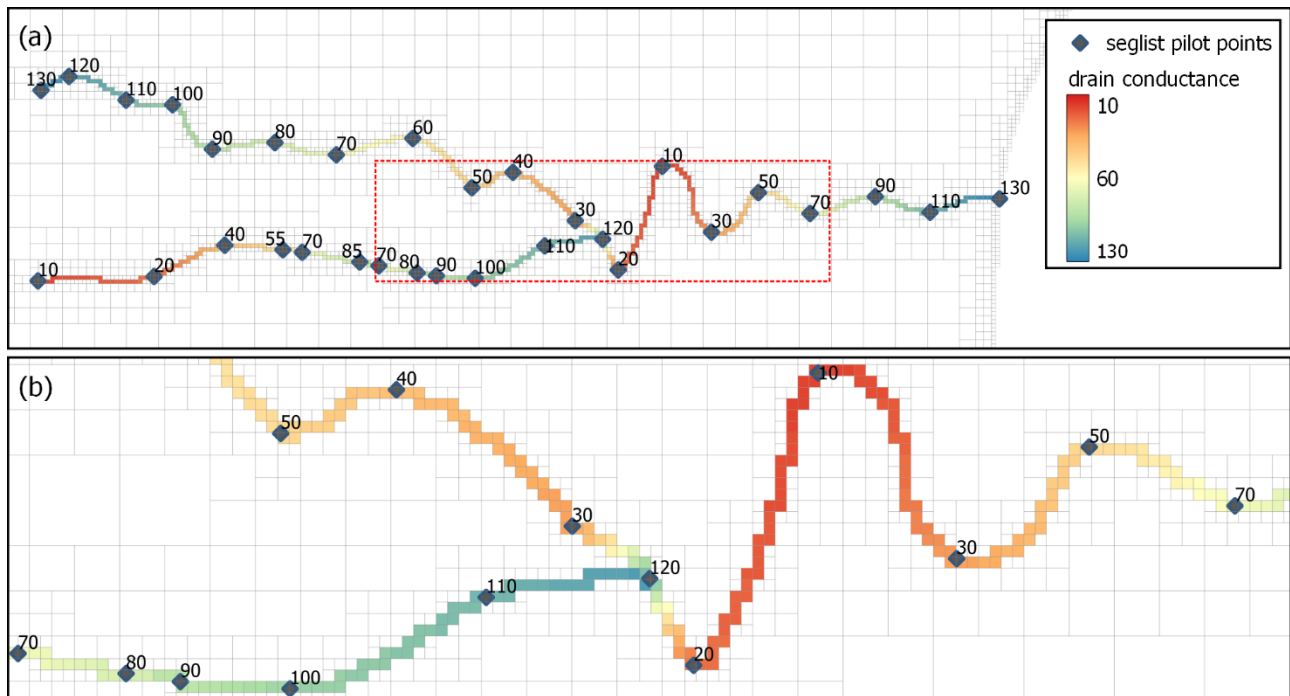


Figure 10 Model drain cells with linearly varying conductance values interpolated from SEGLISTS; (labels indicate SEGLIST pilot point conductance values).

Great job, you have completed the interpolation to linear features tutorial. We hope you found it useful! Why don't you experiment with using the *midpoint* linkage type (see step 45) to see how interpolation outcomes differ?

All files generated up to this point can be found in the folder *t4*. The folder also contains batch files (*.bat) to automate the model display steps.



gmdsi.org

CRICOS NO 00114A

BHP



Flinders
UNIVERSITY



NATIONAL CENTRE FOR
GROUNDWATER
RESEARCH AND TRAINING

RioTinto