A Synthetic Case

Calibration

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

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

This document is part of series of tutorials which demonstrate workflows for parameter estimation and uncertainty analysis with the PEST/PEST++ suites. These are not the only (or necessarily the best) workflows; their purpose is to take the reader through the fundamentals of how to accomplish common tasks whilst also providing insights in how to apply the outcomes.

The PEST roadmaps are a useful complement to this series of tutorials. Going through **Roadmap 10: Model Calibration** and **Roadmap 7: A Pilot Points Workflow** prior to commencing the current tutorial is recommended. The PEST input dataset used in the current tutorial makes use of utilities from the PEST suite such as PLPROC and OLPROC. Familiarity with these utilities is not a prerequisite for completing the current tutorial; however, it is recommended. Other GMDSI tutorials provide introductions to PLPROC and OLPROC.

The present document is a tutorial on how to prepare a PEST input dataset and then calibrate a (very) simple model. Several different regularisation options are demonstrated to highlight the different outcomes of using each of them. This same model and PEST input dataset will provide the basis for subsequent tutorials in this series.

The current tutorial is split into roughly three parts. Chapters 1 and 2 provide a bit of background on the case and model setup; these chapters place no requirements on you (the reader) other than to actually read them. Chapter 3 describes how the PEST input dataset was set up. You are not required to repeat these steps, but they are described in case you wish to do so. Chapter 4 is where the tutorial kicks-off in earnest. This is where you are expected to follow along by fine tuning, running, and assessing the history-matching process.

Software Executables

A number of programs are used throughout this tutorial. These include MODFLOW 6, PEST_HP and several utilities from the PEST suite. For the purposes of the tutorial, executable versions of these programs have been placed in relevant folders (these are **.exe* files). This is generally not recommended practice, for data files and executable files should be kept separate!

Preferably, executables should be located in a folder that is cited in your computer's PATH environment variable. Doing so allows you to run them from a command prompt open to any other folder without having to include the full path to these executables in the command to run them

1.1 Background

The case being modelled is entirely synthetic. It is the authors' experience that building synthetic models that look like "the real thing" never really works well. However the synthetic nature of a didactic model does not detract from the lessons that it teaches. Some design decisions have been taken to highlight the application of specific tools or techniques at the expense of sometimes making the case "un-realistic". So please bear with us, dear reader, if sometimes things look a bit weird!

Our imaginary site looks something like Figure 1. With an area covering roughly 10x10 km, it is bordered to the east by a large river and to the west by gradually rising hills. Several streams traverse the area, flowing from west to east. These streams are perennial. Stream flow is maintained by groundwater. Let us imagine that these streams are one of the last remaining habitats of a particular species of frog which needs an aquatic environment all year round. Let us accept that it is in humanity's best interests to keep this species of frog alive.

There are two large plots of irrigated agricultural land in the area (see land use 2 and 3 in Figure 1). Historically, these are the two principal groundwater users in the area. The nearby town of

Makebelievesville needs to expand its water supply through groundwater abstraction. Land access and existing infrastructure limit possible sites for a wellfield. Of these, the preferred site is close to one of the streams (the red star in Figure 1). Proposed abstraction rates are a constant 2,000 m^3/d. The proximity of the proposed extraction well to the stream raises concerns that streamflow may be depleted. After extensive research, ecologists have determined that the frog species requires a minimum flow rate of 6,000 m^3/d to survive. If proposed abstraction reduces groundwater discharge to the streams to less than this value, this is construed as “a bad thing” from an environmental management point of view.

Outcrop geology in the area is dominated by relatively permeable sands which reach up to 30 m in thickness near the hills in the west and thin to around 3 m near the river in the east. Near the streams, this formation has been eroded and replaced by alluvium deposits. Underlying the upper sand formation is a ~10 m thick layer of clayey deposits. The composition of this formation is variable, but is generally believed to be comprised of low permeability material. Finally, a 70 m thick layer of fine sands underly the clay layer. Impermeable bedrock forms the bottom of the aquifer system.

The system is understood to be a multilayered aquifer system, with the clay layer behaving as an aquitard, separating the shallow unconfined aquifer from the deep confined aquifer. Groundwater gradient is generally from west to east, discharging towards the river in the unconfined aquifer. Groundwater from the unconfined shallow aquifer feeds the streams.

The unconfined aquifer is recharged by rainfall and irrigation return flow. Lateral flow from the west contributes recharge to both aquifers.

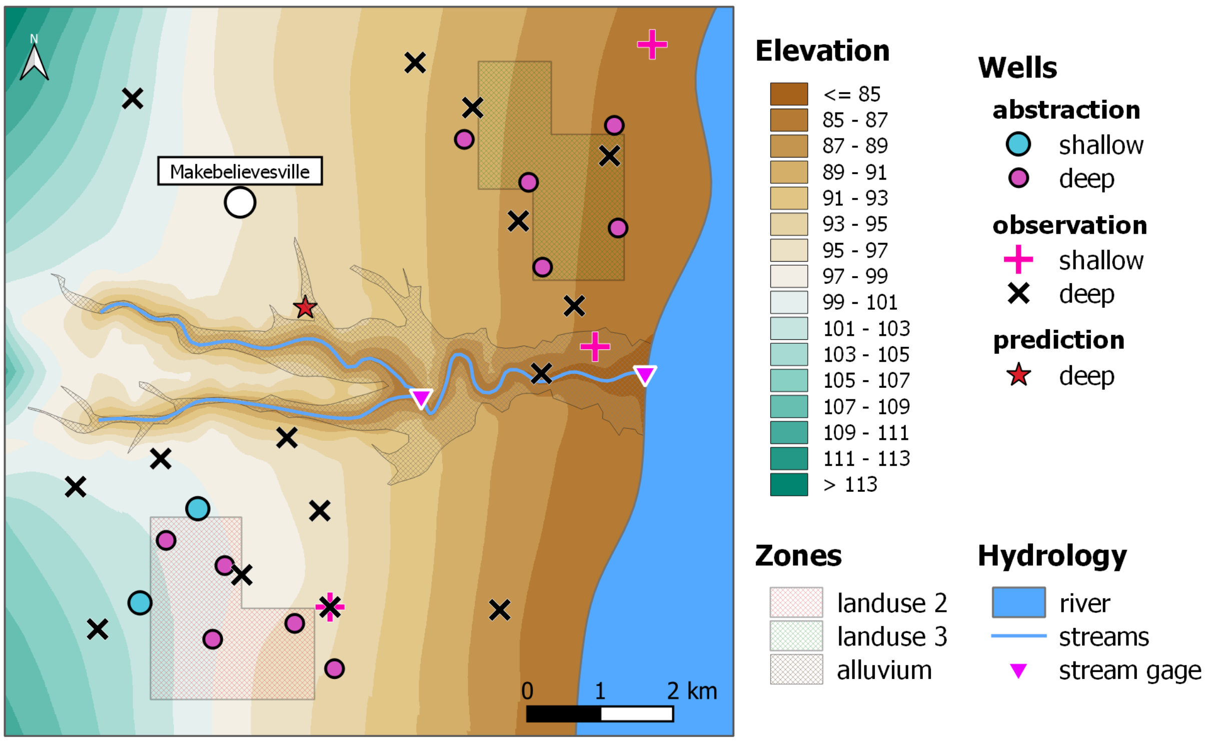


Figure 1 – Layout of the synthetic case and main features of interest.

1.2 The Prediction

A numerical model is developed to assess whether proposed abstraction will reduce groundwater discharge to the streams below the critical limit of 6,000 m^3/d under long-term average conditions (i.e., steady state). It is assumed that future climatic conditions will remain the same as during the historical period (just to keep things simple for the tutorial).

2. THE SIMULATION

2.1 The Simulator

The case described in this document is simulated using a single MODFLOW 6 model. At the time of writing, MODFLOW 6 is the latest generation of the USGS family of MODFLOW simulators. Of primary interest to the present case is its ability to employ an unstructured grid and write outputs to CSV files. Its former characteristic supports grid refinement in the vicinity of the streams.

This series of tutorials demonstrates the use of several utilities which are designed to complement MODFLOW 6's capabilities. However, the workflow described herein can easily be adapted to other simulators.

2.2 The Groundwater Model

Control Settings

MODFLOW 6 simulation control settings are not described in detail in this tutorial as they are not particularly relevant. However, there is one setting which deserves special mention. This is the CONTINUE flag. It should always be activated when running MODFLOW 6 under the control of PEST or PEST++.

Inclusion of the CONTINUE flag in the simulation NAM file (*mfsim.nam*), instructs MODFLOW 6 that simulation should continue even if the solution does not converge during a particular time step. If a model is numerically temperamental, it may experience solution convergence difficulties when PEST updates certain parameter values. Without the CONTINUE flag, MODFLOW will output an error message and halt the simulation when this occurs. This will cause PEST to cease execution; the inversion process with thereby come to a halt.

Sometimes convergence issues can suggest inadequacies in a conceptual model, or in setup of a numerical model. These should be investigated. On other occasions they may be isolated incidents. By including the CONTINUE flag, MODFLOW 6 will continue its simulation to the end, regardless of failure to achieve solution convergence during any particular time step. While this may affect the quality of finite-difference derivatives, PEST execution will not be interrupted.

Model Geometry

The model employs a rectangular quadtree-refined grid (created using USGS software GRIDGEN) to represent the 3 layered aquifer system displayed in Figure 2. The layer 1 (the upper layer) 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. Over most of the model domain cell dimensions are 250m x 250m; however, cells are quadtree refined to 25 x 25 m along streams and rivers. The top elevation of each cell in the upper layer is set to the topographic elevation. The bottoms of layers 1, 2 and 3 are set at 80 m, 70 m, and 0 m, respectively. Layers 2 and 3 have uniform thicknesses.

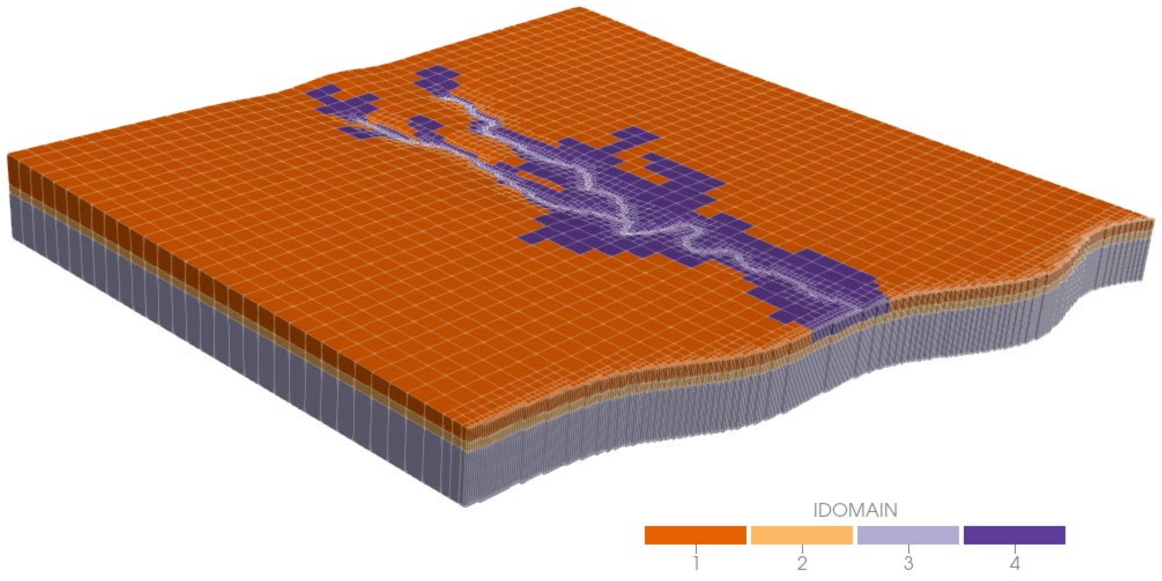


Figure 2 Three-dimensional view of the model grid; zones are defined by IDOMAIN value. Vertical scale is exaggerated x10.

Time Discretisation

Two steady-state stress periods are defined They represent:

1. historical conditions of the system and
2. long-term average conditions with additional proposed abstraction for public supply.

Hence both the history-matching period and the prediction period are included in the same model run. While this is not a requirement for use of a model with PEST , it is convenient for when we later assess predictive uncertainty. By including the prediction as part of the history-matching simulation, sensitivities of the prediction to changes in model parameters are readily calculated. (Predictive and parameter uncertainty are addressed in subsequent GMDSI tutorials).

A disadvantage of this strategy, however, is that joint simulation of both the historical d and prediction periods in the same model run lengthens the model's run time. As the current model takes less than a second to run, the additional computational burden incurred by simulation of future conditions is not significant. This may not be the case for real-world cases where the numerical cost of running the model into the future may be considerable. In this case the, predictive sensitivities should be calculated in a PEST run which is dedicated to this purpose.

Boundary Conditions

The connection to the river along the eastern boundary of the model is represented using the constant head (CHD) package. This is applied only in layer 1. The head is set to 83 m (the river stage) along the entire length of this boundary. Underlying cells in layers 2 and 3 are no-flow boundaries.

The system receives groundwater inflow from the hills to the west. These flows are simulated using the MODFLOW 6 general head boundaries (GHB) package. GHBs are assigned to cells along the western boundary in layers 1 and 3. The head in the groundwater system to the west is set to 102 m. GHB conductances in layers 1 and 3 are independent of each other; they are parameterized using pilot points (see Chapter 2.3).

Streams crossing the system are represented using the river (RIV) package. River stage and bottom are set to 0.5 m and 1 m below topographic elevation, respectively. Riverbed conductance is parameterized using pilot points (see Chapter 2.3).

To preclude hydraulic heads rising above the topographic surface, MODFLOW 6 drains (using the DRN package) are assigned to all cells in layer 1. Drain conductance is simply set very high. This strategy allows water to find its own points of escape from the groundwater system to the surface – something that, of course, occurs in nature.

Model boundaries along the north and south of the domain are no-flow in all layers. All model boundaries in the aquitard (layer 2) are no-flow.

Recharge and EVT

Recharge (RCH) and evapotranspiration (EVT) throughout the model domain are assigned according to the three land use zones depicted in Figure 1; these are comprised of two agricultural zones (land use 2 and 3 in Figure 1) and a "natural vegetation" zone (everywhere else).

For simplicity, EVT rates and extinction depths are constant throughout the model domain and not included as PEST-adjustable parameters. The RCH package is used to apply rainfall-based recharge across the entire model domain. The rate of recharge rate is described using pilot points; recharge values ascribed to these pilot points are adjusted by PEST. Additional recharge is assigned to each of the agricultural land use zones; these rates are constant and are not adjusted by PEST (for the purpose of the tutorial, we simply assume that we are certain of this value).

Abstraction

Groundwater abstraction is simulated using the MODFLOW 6 well (WEL) package. Values are assigned to each well according to "known" abstraction rates. Pumping rates are not adjusted by PEST. During predictive simulations, an additional single WEL boundary condition is assigned at the proposed well location with a constant abstraction rate of 2,000 m^3/d.

2.3 Spatial Parameterisation

Spatial parameterisation is accomplished using pilot points; PEST assigns values to pilot points, and these values are spatially interpolated to cells of the model domain. Spatial interpolation from pilot points to the model grid is accomplished using PLPROC, a "parameter list processor" supplied with the PEST suite. PLPROC was written specifically to facilitate flexible pilot points parameterisation of two- and three-dimensional model domains. (See the relevant GMDSI Tutorials on PLPROC and pilot points and on interpolation along linear features).

For the present tutorial, all PLPROC scripts and associated files can be found in a folder named *runmodel\preproc* within the overall tutorial folder. Their details will not be discussed here; please see the abovementioned GMDSI tutorials for information on how to set up PLPROC scripts.

Hydraulic Conductivity and Recharge Rate

Pilot points are used as a parameterisation device for horizontal hydraulic conductivity (Kh) in all three layers. Vertical hydraulic conductivity (Kv) is assumed be an order of magnitude lower than Kh. The same pilot points are used to parameterize recharge (RCH) in the top layer.

Pilot points are listed in file *ppoints.dat* in the *runmodel\preproc* folder within the tutorial directory. In the same folder you will find the PLPROC scripts which handle spatial interpolation of Kh and RCH to the model grid; these files are named *plproc.dat* and *plproc_rch.dat* respectively.

In layer 1, pilot points are separated into two zones; the assignment of each to a particular zone depends on whether it lies with the alluvium zone or not. Pilot point placement is roughly in accordance with the following principles:

(a) Placement of pilot points extends out to the model boundaries (albeit with a lower spatial density as boundaries are approached);

(b) Pilot points are placed between observation wells and
    i. other observation wells in the up or down-gradient direction,
    ii. pumping wells, and
    iii. outflow/inflow boundaries;
(c) A greater spatial density of pilot points is employed where there is a greater density of observation points;
(d) A minimum spatial density of pilot points is preserved everywhere else.

GHB and RIV Conductance

As described above, the MODFLOW 6 model employs the GHB package along the western boundary of layers 1 and 3. At the same time, the RIV package simulates the interaction between groundwater and the streams that traverse the system. Both of these packages require a conductance term that controls the rate of flow across the boundary.

Conductances of these linear features are also parameterized using pilot points to allow for variation of this property along their lengths. Conductance of the GHB is parameterized using traditional pilot points. The same pilot points are used for both of layer 1 and layer 3; however the values assigned to conductances in each layer are distinct. In contrast, conductance of the RIV boundary is accomplished using PLPROC's SEGLIST functionality, in order to respect the geometry of polylinear features during interpolation. (See the relevant GMDSI Tutorial on interpolation along linear features).

PLPROC scripts named *plproc_ghb.dat* and *plproc_stream.dat* handle spatial interpolation of GHB and RIV conductance parameters to pertinent model cells. Respective pilot points are defined in files *ghbpp.dat* and *streampp.dat*, should you wish to inspect them. All of these can be found in the *runmodel\preproc* folder.

2.4 The Observations

The history-matching dataset is comprised of long-term average hydraulic heads measured at boreholes screened in either the shallow or the deep aquifer. At one location there is a nested piezometer with head values measured in both the shallow and deep layers. Long-term average stream flows are measured at two stream gauges, as shown in Figure 1. All of these "measurements" which comprise the calibration dataset were generated using the model described above populated with a random parameter field. Another GMDSI tutorial will demonstrate how to accomplish this.

Simulated outputs of interest are recorded using MODFLOW 6's observation (OBS) package. Hydraulic head time-series at the locations and layers of observation wells are recorded at every simulated time-step (i.e., at the end of each steady state stress period). For the sake of simplicity, each of our make-believe observation wells coincides with the centre of a model cell. These are compared to "measured" hydraulic heads during the history-matching process. The OLPROC utility is used to postprocess MODFLOW 6 outputs whenever the model is run and record them in a separate file. (This is especially important during transient calibration when model-generated quantities have no observed counterparts during some model time steps.) OLPROC also writes PEST instruction files to read its own output files; see Chapter 3.4.

The difference in head between the shallow and the deep aquifer at the site of the nested piezometer carries useful information on the local vertical hydraulic conductivity of the aquitard layer. OLPROC can be used to postprocess model outputs and generate "secondary observations" of head differences between two relevant observation points; these can then be matched to observed head differences between the same points (See GMDSI OLPROC tutorial for details on how to accomplish this.)

Simulated exchange of water between the aquifer and the stream at each time step is recorded for two stream sections. These are:

(1) tributary reaches upstream of the first gauge, and
(2) the downstream reach between the two gauges.

However, measured data at the downstream gauge is in fact the accumulated flow for all reaches. Comparison of model outputs to measured data during the history-matching process requires that the groundwater-surface water exchange rates for the two reaches be added. (For the sake of simplicity, we assume that stream residence time is very short). As a model postprocessor, this task can be accomplished by OLPROC. (See the GMDSI OLPROC tutorial for details of how to accomplish this.)

No noise was added to “measurements” comprising the calibration dataset. However, for the purposes of the tutorial, we shall assume that head measurements have an associated error of +/-0.1 m. We shall also assume that stream flow rates have an associated error that is equal to +/-10% of the measured value; that is, we assume that error scales with flow rate.

Our model setup requires that simulated flow out of DRN cells in the top model layer is recorded by MODFLOW 6. These outflows are not compared explicitly to any measurements. However, through the use of penalty functions they could be used to assimilate “soft data” (such as knowledge that the area is not consistently flooded) in order to guide the history-matching process towards rejection of parameters that promote aberrant model behaviour. The MODFLOW 6 model is also set up to record flows through other boundary conditions (GHB, WEL, RCH, EVT and CHD). However, these are merely a convenience; they are not used for history-matching or prediction.

3. THE PEST INPUT DATASET

A PEST input dataset is comprised of three file types:

- template files,
- instruction files, and a
- PEST control file.

Template files must usually be constructed by the user. Instruction files and the PEST control file can be partially constructed using utilities from the PEST suite. However, some user input and file manipulation are also required. Construction of these files is touched upon in other GMDSI Tutorials (see PLPROC: Basics and OLPROC); it is also described in the PEST manual and other PEST tutorials.

Most groundwater modelling graphical user interfaces (GUI's) support construction of PEST input datasets of varying complexity. In many cases, PEST setup achieved in this way is fit for purpose. However, no GUI is able to cater for every conceivable history-matching context. Hence there are many contexts in which a user may need to construct his/her own PEST input dataset. The following chapters of this tutorial provide some guidance on how to do this.

The current chapter follows a similar structure to that of the PEST uncertainty tutorial document. Brief descriptions of the steps required for construction of a PEST input dataset are provided, along with the final result. There is no need for you to carry out any of these steps if you do not wish to. The hands-on part of the tutorial begins in Chapter 4.

We assume that you are at least partly familiar with PEST, and with programs that comprise its utility support suite. Other GMDSI tutorials describe use of utilities that facilitate the construction of a PEST input dataset in greater detail.

3.1 Directory Structure

The number of files which comprise a PEST input dataset can become quite large. If MODFLOW's input and output files, and those required by its pre- and postprocessors, are added to these, then keeping track of all of them can become an onerous task. Their number and complexity can also make error-tracking difficult.

Having organised data and a logical directory structure can make life easier. On the other hand, keeping track of relative path references between the various files and components which comprise a PEST input dataset can also become complex. In the end, there is no "correct" way to organize a directory structure for a PEST input dataset. The structure that you choose is likely to emerge from personal preferences and/or project-specific constraints.

The current tutorial uses the directory structure depicted in Figure 3. Each blue rectangle represents a folder:

- **runmodel**: contains all folders which comprise the composite groundwater model and all model pre- and post-processing steps.
  - **model**: contains all MODFLOW 6 input and output files
    - **obs:** contains all model-generated CSV files.
  - **preproc**: contains all PLPROC script and pilot point files that are used for model parameterisation.
  - **postproc**: contains all OLPROC scripts and measurement SSF files that are used to construct and postprocess model-simulated observations.

- **pest:** contains PEST control files and PEST-generated files, along with the model batch file referenced in the PEST control file.
    - **template:** contains all parameter template (*.*tpl*) files referenced in the PEST control file (see Chapter 3.2).
    - **instruction:** contains all observation instruction (*.*ins*) files referenced in the PEST control file (see Chapter 3.4).

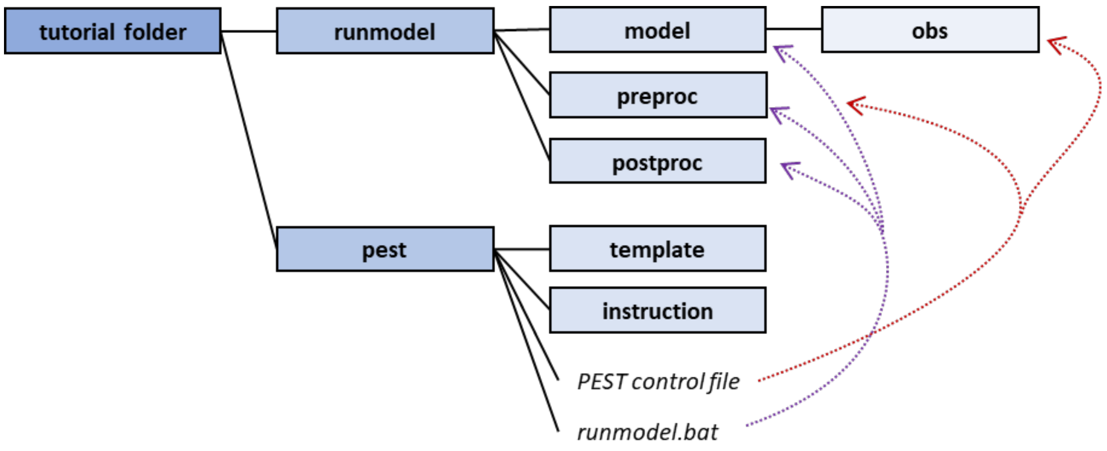


Figure 3 – Directory structure for model and PEST input datasets used in the current tutorial. Each blue rectangle represents a folder. Dotted purple and red lines indicated relative path references in the PEST control file and in the model batch file to files in respective folders.

3.2 Building Template Files

Whenever PEST runs a model, it must first write parameter values to model input files. PEST requires template files in order to know how to write model input files in ways that the model expects. By convention, template files are provided with an extension of "*.tpl*".

As described in Chapter 2.3, three sets of pilot points are used to parameterize spatially varying hydraulic properties. A template file is required for each of the pilot point files. You do not need to build a template file as part of this demonstration. They have been provided for you in the folder *pest\template*. As previously described, the pilot point files are named *ppoints.dat, ghbpp.dat* and *streampp.dat.* The corresponding template files are named *ppoints.tpl, ghbpp.tpl* and *streampp.tpl*. Inspect them with a text editor. Note that the first line in all of these template files is "*ptf $*", and that columns that contain parameter values have been replaced with columns that contain parameter names; these are delimited by the "*$*" character. These parameter names will be referenced in the PEST control file; this is described in the next chapter.

If you wish to gain some practice, why not try and build these template files yourself? You can accomplish this relatively easily using any text editor and/or spreadsheet software. Alternatively (and probably better), you can also do so programmatically (using Python for example).

If you do construct them yourself, make sure to use the TEMPCHEK utility to verify that the files are constructed correctly. (You will need to have the *tempchek.exe* executable file in your working folder, or preferably in a folder which is cited in your computer's PATH environment variable.) You can run TEMPCHEK by opening a command line in your working folder and typing the commands (replacing *filename.tpl* with the name of your template file):

```
tempchek filename.tpl
```

3.3 Building a Partial PEST Control File – Parameters

TPL2PST is a utility from the PEST suite; it is described in Part 2 of the PEST manual. TPL2PST is used to build a “partial PEST control file” based on the contents of one or a number of template files. See the PEST manual for details of its use.

When assembling a PEST input dataset for calibration of a complex model, template files are often built first. TPL2PST can read all of these files. It then builds a PEST control file in which the following sections are populated. (Headers are provided for other sections of this file, even though they are empty.)

- control data
- singular value decomposition
- parameter groups
- parameter data
- model input/output

The partial PEST control file that is written by TPL2PST will require manual editing. Furthermore, its other sections must be populated. (As will be demonstrated later in this tutorial, this can be done with the help of PEST support programs such as OLPROC, as well as Groundwater Utilities such as PESTPREP1 and PESTPREP2.) However, because it populates the “control data” section of a PEST control file, and includes in this file all parameters that are cited in one or a number of template files, it provides an easy and less error-prone alternative to manual construction of these sections of a PEST control file.

TPL2PST’s tasks are governed by the contents of an input file. This file is easily prepared using a text editor. A TPL2PST input file named *tpl2pst.in* has been provided in the tutorial folder. You can inspect it using any text editor.

If you wish, go ahead and run TPL2PST to create a partial PEST control file. (You will need to have the *tpl2pst.exe* executable program in your working folder, or preferably in a folder that is cited in your computer’s PATH environment variable.)

1 Open a command line in the *template* folder, type the following command and then pressing <enter>:

```
tpl2pst tpl2pst.in param.pst
```

2 You should see the following text appear on your screen:

```
- file tpl2pst.in read ok.

 - reading file ppoints.tpl...
 - 2092 params found in file ppoints.tpl.

 - reading file streampp.tpl...
 - 31 params found in file streampp.tpl.

 - reading file ghbpp.tpl...
 - 82 params found in file ghbpp.tpl.

 - writing file param.pst...
 - file param.pst written ok.
```

3 You should now have a new file named *param.pst* in your folder. If you inspect it, you will see that it is a PEST control file populated with all of the parameters that are referenced in the *.*tpl* files that were provided with this tutorial All of these parameters have been endowed with the

parameter group names, initial values, bounds and transforms that were provided in the TPL2PST input file *tpl2pst.in*.

4 There is however, one problem. All of the parameters listed in *ppoints.tpl* (which pertain to Kh and RCH for the various layers) have been assigned to the same parameter group. They have also been given the same initial values and upper/lower bounds. See below a snippet from the TPL2PST-generated *param.pst* PEST control file:

```
* parameter data
kx1pp001  log  factor  1.000000   0.1000000  10.00000  changethis   1.0   0.0
kx2pp001  log  factor  1.000000   0.1000000  10.00000  changethis   1.0   0.0
kx3pp001  log  factor  1.000000   0.1000000  10.00000  changethis   1.0   0.0
rchpp001  log  factor  1.000000   0.1000000  10.00000  changethis   1.0   0.0
(…)
ghbc1pp01  log  factor  100.0000  1.0000000E-04  10000.00  ghbc   1.0   0.0
ghbc3pp01  log  factor  100.0000  1.0000000E-04  10000.00  ghbc   1.0   0.0
```

5 The same has occurred for pilot points associated with GHB conductance: pilot point parameters from layer 1 and layer 3 have both been assigned to the same parameter group. It is generally advisable to group parameters both per type and unit (for example layer or zone).

6 For pilot point parameters, this could have been avoided by having distinct pilot point files for each parameter type; however this would have made setting up PLPROC slightly more complicated. Even if this were done, our PEST control file would still need to be edited. It is easy to do this manually. You can copy the “parameter data” section of a PEST control file into a spreadsheet package, or even edit the file directly in a text editor. It is also a relatively simple task to manipulate the contents of a PEST control file using a self-written program.

7 Parameter group names *kx1*, *kx2*, *kx3*, *rch, ghb1* and *ghb3* need to be assigned to respective parameters in the “parameter data” section of the PEST control file and added to the “parameter groups” section of this file. Relevant initial values and upper and lower bounds need to be assigned to all parameters. As the number of parameter groups increases, the NPARGP control variable in the “control data” section of the PEST control file must be updated accordingly. (NPARGP is the number of parameter groups cited in a PEST control file.)

8 Assign the following initial values and bounds to all parameters which belong to respective parameter groups.

Table 3-1 Parameter groups, initial values and bounds assigned in the PEST control file *partial_param2.pst*.

Type	Count	Parameter group	Initial value	Lower bound	Upper bound
Pilot points	523	kx1	5	1.00E-03	1.00E+03
	523	kx2	0.1	1.00E-03	1.00E+03
	523	kx3	5	1.00E-03	1.00E+03
	523	rch	2.74E-04	9.13E-05	8.23E-04
	31	strc	100	1.00E-04	1.00E+04
	41	ghb1	100	1.00E-04	1.00E+04
	41	ghb3	100	1.00E-04	1.00E+04
Total	**2205**				

9 Note that the order in which parameters are listed in the “parameter data” section of a PEST control file does not matter at this stage. However, it will matter in the future when assigning covariance matrices to groups of parameters for regularisation purposes (Chapter 4.3). To reduce the chance of future mistakes, it is usually good practice to order parameter names in a PEST control file, and group them by parameter group.

10 You can see the outcomes of this work in the PEST control file named *param2.pst* which has been provided in the *pest\template* folder. The parameters discussed above should appear as follows. Values that have been changed are in **bold;** (…) indicate lines in the file which are not shown:

```
* parameter data
kx1pp001   log   factor  5.000000   1.00E-03  1.00E+03  kx1    1.0    0.0
(…)
kx2pp001   log   factor  0.100000   1.00E-03  1.00E+03  kx2    1.0    0.0
(…)
kx3pp001   log   factor  5.000000   1.00E-03  1.00E+03  kx3    1.0    0.0
(…)
rchpp001   log   factor  2.74E-04   9.13E-05  8.23E-04  rch    1.0    0.0
(…)
ghbc1pp01   log   factor  100.0000  1.000000E-04  10000.00  ghb1    1.0    0.0
(…)
ghbc3pp01   log   factor  100.0000  1.000000E-04  10000.00  ghb3    1.0    0.0
```

11 Note that the "parameter groups" section of the PEST control file has also been updated, along with the NPARGP variable in the "control data" section (updated values are depicted in **bold**).

```
pcf
* control data                        NPARGP
restart estimation
 2205        0          7          0         0
 3  0    single  point
 10.0  -3.0  0.3  0.03  10  lamforgive
 10.0  10.0  0.001
 0.1
 50  0.005  4  4  0.005  4
 0  0  0
* singular value decomposition
1
5000  5e-7
0
* parameter groups
kx1    relative   0.015   0.0  switch  2  parabolic
kx2    relative   0.015   0.0  switch  2  parabolic
kx3    relative   0.015   0.0  switch  2  parabolic
rch    relative   0.015   0.0  switch  2  parabolic
strc   relative   0.015   0.0  switch  2  parabolic
ghb1   relative   0.015   0.0  switch  2  parabolic
ghb3   relative   0.015   0.0  switch  2  parabolic
```

12 Once you are satisfied with *param2.pst*, copy it to the *pest* folder and rename the copy as *calib0.pst*.

Because of the directory structure that was adopted for this tutorial, the PEST control file is not housed in the same folder as model inputs or template files. Therefore, we need to add the relative file paths for each of these files to the PEST control file.

13 Open *calib0.pst* in a text editor. Scroll down to the end of the file and inspect the "model input/output" section of this file. It should look something like this:

```
* model input/output
ppoints.tpl    ppoints.dat
streampp.tpl   streampp.dat
ghbpp.tpl      ghbpp.dat
```

14 Each line within this section references two file names. The first is a template file and the second is the corresponding model input file. Template files are housed in the *pest\template* folder. In this case, all of the model input files are pilot point files; these are all housed in the *runmodel\preproc* folder. We need to add the relative folder path (i.e. the path relative to the folder in which *calib0.pst* is stored) in order to inform PEST where each file is located.

15 The *template* folder is a subfolder of that in which *calib0.pst* is stored Therefore, for example, the relative path from *calib0.pst* to *ppoints.tpl* can be written as "*.\template\ ppoints.tpl*".

16 The relative path to the *preproc* folder is slightly more complicated. It needs to navigate one directory (i.e. folder) up, then one directory down into the *runmodel* folder and then down again into the *postproc* folder Thus, for example, the relative path to *ppoints.dat* can be written as "*..\runmodel\preproc\ppoints.dat*". (".." specifies, up one directory level; "." specifies, in the same directory level.)

17 Add relative paths to each file in the "model input/output" section of *calib0.pst*. Once you have done this, it should look something like this:

```
* model input/output
.\template\ppoints.tpl     ..\runmodel\preproc\ppoints.dat
.\template\streampp.tpl    ..\runmodel\preproc\streampp.dat
.\template\ghbpp.tpl       ..\runmodel\preproc\ghbpp.dat
```

Now that the "parameter data" and "parameter groups" sections of the PEST control file are complete, we can proceed to add sections that pertain to observations.

3.4 Building a Partial PEST Control File – Observations

"OLPROC" stands for "observation list processor". Its name complements those of other members of a suite of programs that facilitates use of PEST and PEST++ in conjunction with groundwater models in general, and the MODFLOW family of models in particular. OLPROC's role is to postprocess model output time series to enable their comparison with field measurements by PEST. It also facilitates the construction of a PEST input dataset (control file and instruction files) in which OLPROC itself is run as a model output postprocessor. These two modes in which OLPROC operates are referred to as "postprocessor mode" and "constructor mode". See the GMDSI Tutorial on how to use OLPROC to construct a PEST control file and postprocess model outputs as part of a PEST controlled model run.

As was described in Chapter 2.4, OLPROC is used to postprocess output files generated by MODFLOW 6 in order to facilitate comparison with measured data by PEST. Two OLPROC script files named *olproc.in* and *olproc-pred.in* have been provided for you; these can be found in the *runmodel\postproc* folder. These OLPROC input files are used to construct partial PEST control files based on OLPROC-postprocessed model outputs pertaining to the history-matching and prediction periods, respectively.

The same folder contains site sample files (i.e. SSF files) which house "measured" observation data. These files are named *obs-heads1.ssf, obs-heads3.ssf, obs-headdiff.ssf* and *obs-stream.ssf*. As was described previously, these "measured" observations were generated by the model when supplied with a random parameter set. An additional SSF file named *pred-stream.ssf* provides dummy measurements for the "prediction observation".

When OLPROC runs the *olproc.in* or *olproc-pred.in* script in constructor mode, it writes a partial PEST control file; these are named *partial1.pst* and *partial2.pst*, respectively. It also generates a set of instruction (**.ins*) files. The instruction files must be copied to the *pest\instruction* folder. Observation data recorded in the two partial PEST control files must be amalgamated into a single PEST control file. These tasks have already been accomplished for you. Should you wish to repeat them yourself, do the following:

18 Open a command line window in the *runmodel\postproc* folder. Run OLPROC in constructor mode with the first script by typing the following command and then pressing <enter>:

```
olproc olproc.in 0
```

19 You should see the following text on your screen:

```
- file olproc.in read ok.
- file ../model/obs/head_obs.csv read ok.
- file ../model/obs/stream_flow.csv read ok.
- file obs-heads1.ssf read ok.
- file obs-heads3.ssf read ok.
- file obs-stream.ssf read ok.
- file obs-headdiff.ssf read ok.
- model observation equations evaluated ok.
- file .\obs_files\o_obs-heads1_ssf.ssf written ok.
- file m_obs-heads1_ssf.ins written ok.
- file m_obs-heads1_ssf.ssf written ok.
- file .\obs_files\o_obs-heads3_ssf.ssf written ok.
- file m_obs-heads3_ssf.ins written ok.
- file m_obs-heads3_ssf.ssf written ok.
- file .\obs_files\o_obs-stream_ssf.ssf written ok.
- file m_obs-stream_ssf.ins written ok.
- file m_obs-stream_ssf.ssf written ok.
- file .\obs_files\o_obs-headdiff_ssf.ssf written ok.
- file m_obs-headdiff_ssf.ins written ok.
- file m_obs-headdiff_ssf.ssf written ok.
- file partial1.pst written ok.

  Number of observations [NOBS]         = 21
  Number of observation groups [NOBSGP] = 4
  Number of instruction files [NINSFLE] = 4
```

20 Note the last three lines. Write down the values of NOBS, NOBSGP and NINSFLE (recorded in **bold** above). You will need these later.

21 Now, run OLPROC with the second script by typing *olproc olproc-pred.in 0* and then pressing <enter>. Your screen should show the following text:

```
- file olproc-pred.in read ok.
- file ../model/obs/stream_flow.csv read ok.
- file pred-stream.ssf read ok.
- model observation equations evaluated ok.
- file .\obs_files\o_pred-stream_ssf.ssf written ok.
- file m_pred-stream_ssf.ins written ok.
- file m_pred-stream_ssf.ssf written ok.
- file partial2.pst written ok.

  Number of observations [NOBS]         = 2
  Number of observation groups [NOBSGP] = 1
  Number of instruction files [NINSFLE] = 1
```

22 Again, note the last three lines. Write down the values for NOBS, NOBSGP and NINSFLE. You will need these later.

23 Inspect the *runmodel\postproc* folder. You should see two sets of SSF and instruction (i.e. *.ins*) files, all beginning with the prefix "*m_*". As you may recall, the SSF files are written by OLPROC when run as a model-postprocessor. Hence, from PEST's point of view, they are model output files. They contain MODFLOW 6 outputs, time-interpolated (by OLPROC) to the times at which measurements were made. These measurements are recorded in the OLPROC-produced PEST control file. The instruction files recorded by OLPROC instruct PEST how to read the output files which it records as a MODFLOW 6 postprocessor.

24 You should also see two PEST control files named *partial1.pst* and *partial2.pst*. These are partial PEST control files. They contain the "observations data" and "observation groups" sections of a full PEST control file. In the present case these need to be merged into the final PEST control file.

25 Copy (or move) all the instruction files to the *pest\instruction* folder Leave all the other files where they are in the *runmodel\postproc* folder.

Now we can proceed to assembling the full PEST control file using *calib0.pst* (in the *pest* folder) and the two partial control files *partial1.pst* and *partial2.pst* which OLPROC has just generated.

Observation Groups and Data

26 Open *calib0.pst, partial1.pst* and *partial2.pst* in your favourite text editor.

27 Copy all of the lines from the "observation groups" and "observation data" sections of *partial1.pst* into the corresponding section of *calib0.pst.* Do the same for *partial2.pst*.

28 Once finished, the "observation groups" and "observation data" sections of file *calib0.pst* should look something like this:

```
* observation groups
heads1
heads3
streams
headiff
pstreams
* observation data
 h1-3892_1             89.8431300            10.00000          heads1
 h1-1664_1             84.6600869            10.00000          heads1
 h1-162_1              85.4854459            10.00000          heads1
 h3-3521_1             94.8569409            10.00000          heads3
 h3-3988_1             94.8417582            10.00000          heads3
 h3-3417_1             91.6251609            10.00000          heads3
 h3-3829_1             90.0631982            10.00000          heads3
 h3-3357_1             88.3131842            10.00000          heads3
 h3-3594_1             89.0843112            10.00000          heads3
 h3-3892_1             89.7394028            10.00000          heads3
 h3-1152_1             84.7964916            10.00000          heads3
 h3-313_1              98.4441215            10.00000          heads3
 h3-209_1              90.8085525            10.00000          heads3
 h3-331_1              88.2698626            10.00000          heads3
 h3-707_1              86.5253676            10.00000          heads3
 h3-3956_1             87.4864299            10.00000          heads3
 h3-2270_1             84.6029367            10.00000          heads3
 h3-521_1              85.6793397            10.00000          heads3
 trib_1                -8385.72726          1.1925038E-03      streams
 total_1               -11233.4024          8.9020303E-04      streams
 dh3892_1              0.103727200           100.0000          headiff
 p-trib_1             -6500.000               0.00000          pstreams
 p-total_1            -6500.000               0.00000          pstreams
```

Model Input/Output

29 Inspect the “model input/output” section of each of the control files. The *calib0.pst* PEST control file already references template and model input file pairs. The *param1.pst* and *param2.pst* control files only reference instruction and model output file pairs. Copy the lines from *partial1.pst* and *partial2.pst* and insert them below the existing template and model input file pairs in *calib0.pst.*

30 Update the relative path names for each of the instruction and model output files. Once complete, the “model input/output” section of file *calib0.pst* should look something like this (new lines are in **bold**):

```
* model input/output
.\template\ppoints.tpl                 ..\runmodel\preproc\ppoints.dat
.\template\streampp.tpl                ..\runmodel\preproc\streampp.dat
.\template\ghbpp.tpl                   ..\runmodel\preproc\ghbpp.dat
.\instruction\m_pred-stream_ssf.ins    ..\runmodel\postproc\m_pred-stream_ssf.ssf
.\instruction\m_obs-heads1_ssf.ins     ..\runmodel\postproc\m_obs-heads1_ssf.ssf
.\instruction\m_obs-heads3_ssf.ins     ..\runmodel\postproc\m_obs-heads3_ssf.ssf
.\instruction\m_obs-stream_ssf.ins     ..\runmodel\postproc\m_obs-stream_ssf.ssf
.\instruction\m_obs-headdiff_ssf.ins  ..\runmodel\postproc\m_obs-headdiff_ssf.ssf
```

Control Data section

31 Lastly, we need to update the NOBS, NOBSGP and NINSFLE variables in the “control data” section of *calib0.pst*.

32 Inspect the “control data” section of *calib0.pst* (it is right at the top). It should look similar to this. (You should have “0” instead of **NOBS**, **NOBSGP** and **NINSFLE** in your file.)

```
* control data
restart estimation
 2205         NOBS         7         0         NOBSGP
 3  NINSFLE    single  point
 10.0  -3.0  0.3  0.03  10  lamforgive
 10.0  10.0  0.001
 0.1
 50  0.005  4  4  0.005  4
 0  0  0
```

33 Recall the values that OLPROC wrote to the screen in steps 20 and 22. Replace NOBS, NOBSGP and NINSFLE with the total number of observations (NOBS), observation groups (NOBSGP) and instruction files (NINSFLE). (If you did not record OLPROC's screen outputs, you can simply count the number of lines in the "observation data" section, the "observation groups" section and number of instruction files in the "model input/output" section of *calib0.pst*.)

34 The "control data" section should now look something like this:

```
* control data
restart estimation
2205      23       7       0       5
3  5    single  point
10.0  -3.0  0.3  0.03  10  lamforgive
30.0  30.0  0.001
0.1
50  0.005  4  4  0.005  4
0  0  0
```

35 At this point all parameters and observations should be set up in the PEST control file. Run PESTCHEK to verify the integrity of the control file. To do this, open the command line in the *pest* folder and type *pestchek calib0.pst* and then press <enter>. If all went well, you should see the following message on your screen:

```
Errors ----->
No errors encountered.

Warnings ----->
MAXSING in the singular value decomposition section is greater than the
  number of adjustable parameters.
```

3.5 The "Model Run" File

What PEST sees as "the model" is actually the composite of the MODFLOW 6 groundwater flow simulation and the pre- and postprocessing steps carried out by PLRPOC and OLPROC. PEST runs this "composite model" from a batch file referenced in the "model command line" section of the PEST control file.

An example of such a file named *runmodel.bat* has been provided for you in the *pest* folder. Inspect it by opening it in a text editor. It should look something like this:

```
@echo off
Rem ###################################
Rem Some intermediate files are deleted as a precaution.
Rem ###################################
cd %~dp0..\runmodel\model\obs
del /s *.csv

Rem ###################################
Rem PLPROC assigns parameters from ppoints
Rem ###################################
cd %~dp0..\runmodel\preproc\
plproc plproc.dat
plproc plproc_stream.dat
plproc plproc_ghb.dat
plproc plproc_rch.dat

Rem ###################################
Rem The MF6 model is run
Rem ###################################
cd %~dp0..\runmodel\model\
mf6 mfsim.nam

Rem ###################################
Rem The observations are postprocessed
Rem ###################################
cd  %~dp0..\runmodel\postproc\
olproc olproc.in 1
olproc olproc-pred.in 1

Rem ###################################
Rem Return the command line to the working folder
Rem ###################################
cd %~dp0.\
```

Lines beginning with "Rem" are comments and are ignored when being processed by your computer's operating system. Lines with "*cd %~dp0..*" serve to navigate the system command focus to the relevant folder using relative path names.

This batch file starts by deleting all the CSV files in the *runmodel\model\obs* folder. These are the MODFLOW 6 OBS package output files. By deleting them at the start of the model run, we avoid the possibility of files left over from a previous run being accessed by OLPROC should MODFLOW 6 not run for some reason. After deleting the CSV files, the batch file runs each of the PLPROC scripts to parameterize the model using pilot points. Next, the MODFLOW 6 simulation is run, followed by OLPROC in post processor mode for each of the OLPROC scripts.

36 To verify whether the batch file works, open a command line in the *pest* folder, type the following command and then press <enter>:

```
runmodel.bat
```

37 Outputs from PLPROC, MODFLOW 6 and OLPROC should scroll down the screen. The scrolling then stops. Double check to see if there are any error messages. If not, everything appears to be working as intended.

We are now ready to run PEST. All files produced up to this point can be found in the *pest* and *runmodel* folders provided with the tutorial. These form the starting point for the next chapter.

4. FINE-TUNING THE PEST SETUP

The core of the PEST input dataset has now been assembled. However a few more steps are still required to fine-tune PEST setup before launching it in order to calibrate the model. These include adjusting observation weights so that no single observation group dominates the objective function and adding regularisation in the form of preferred parameter values and constraints on spatial covariance.

4.1 PEST – Run the Model Once

Before going any further, let us make sure that PEST is able to run the model and read all of its output files. To do this, we will get PEST to run the model once.

Set NOPTMAX=0

38 Make a copy of *calib0.pst*;name it *calib1.pst*.

39 Open the PEST control file *calib1.pst* in a text editor. Change the NOPTMAX variable in the “control data” section from 50 to 0 and then save the file. This instructs PEST to carry out a single model run, calculate the objective function, and then cease execution. The “control data” section of *calib1.pst* should now look like this:

```
* control data
restart estimation
2205      23       7       0       5
3  5    single  point
10.0  -3.0  0.3  0.03  10  lamforgive
30.0  30.0  0.001
0.1
0  0.005  4  4  0.005  4
0  0  0
           NOPTMAX
```

PEST_HP

We will be using PEST_HP in the current tutorial. PEST_HP is a version of PEST/BEOPEST whose performance is enhanced in highly parallelized computing environments. Some other aspects of its inversion algorithm are also superior to those of other versions of PEST. If possible, use of PEST_HP is recommended over that of PEST/BEOPEST. While some (rarely used) features of PEST have been removed from PEST_HP, its new features more than compensate for these omissions. See the PEST_HP manual for details.

Two executable versions of PEST_HP are supplied. These are named *pest_hp.exe* and *pest_hp_mkl.exe*. Use of the latter is recommended, as it takes advantage of the Intel Maths Kernel Library to accelerate some matrix intensive operations such as singular value decomposition. If you use *pest_hp_mkl.exe* then a DLL (i.e., a dynamic linked library) named *libiomp5md.dll* must be situated in the folder from which you run *pest_hp_mkl.exe.* Alternatively, it must reside in a folder that is accessible through your computer’s PATH environment variable. Both of these files have been provided in the tutorial folder.

PEST_HP parallelizes model runs using a single “manager” that communicates with multiple “agents”. The manager undertakes numerical tasks required by the inversion process. It also supervises the distribution of model runs to various agents. Agents carry out the model runs; that is, they issue the system command that runs the model. After the completion of each model run, they send model outcomes back to PEST.

Each agent must operate in a separate folder. This folder must contain a copy of the PEST control file. Setup is easiest if each folder also contains copies of all template, instruction and model batch files. So, for the current tutorial, each "agent folder" must contain a copy of the *pest* folder and the *runmodel* folder. This will be discussed in more detail later in the tutorial.

Although PEST_HP works best when it has many parallel agents at its disposal, there is no lower limit to how few agents it can use. To run the model once, we will use only one agent. Later in the tutorial we will calibrate the model using several agents.

Prepare the PEST_HP Manager

40 Open a command line in the *pest* folder and instruct the PEST_HP manager to run using *calib1.pst* by typing the following command and then pressing <enter>:

```
pest_hp_mkl calib1.pst /h :4004
```

41 The following should appear on the screen:

```
PEST is running in parameter estimation mode.

 PEST run record: case calib1
 (See file calib1.rec for full details.)

 Model command line:
runmodel.bat

 RUNNING MODEL FOR FIRST TIME .....
    Running model 1 time....
    Waiting for at least one agent to appear....
```

42 This command executes *pest_hp_mkl.exe* using the PEST control file *calib1.pst*. The port number which PEST_HP will use to communicate with its agents is denoted as *4004*. You could use a different port, as long as it is not being used by any other programs. (Port 4004 is usually free.)

43 Because the PEST_HP manager is running in the *pest* folder, it will record all of its output files in this folder.

At this stage, the PEST_HP manager is running. However it cannot accomplish anything until its agents are also running. So leave the command line window open and proceed to the next step.

Prepare the PEST_HP Agent

As was stated above, each agent must run in its own distinct folder. In principle, there is no reason why one of these agents cannot run in the same folder as a manager; however it must not run in the same folder as another agent. For the sake of the tutorial (and tidy file management) we will create a new folder for our single agent.

44 In the tutorial directory, create a new folder. Name the folder *agent_dir*. In practice you can name it whatever you like. However it is recommended to avoid using spaces in the names of agent folders..

45 In the *agent_dir* folder, create another folder and name it *agent0.*

46 Copy both the *runmodel* and the *pest* folders into the *agent0* folder

47 Open a command line in the *agent_dir\agent0\pest* folder.

48 Start a PEST_HP agent, and tell it to read the *calib1.pst* PEST control file. Do this by typing the following command, and by then pressing <enter>:

```
agent_hp calib1.pst /h %computername%:4004
```

49 Screen outputs from PLPROC, MODFLOW6 and OLPROC should scroll by on the screen. This takes about 1 second. It ends with the text:

```
Model run complete.

 Execution of agent terminated by HP manager.
```

50 Return to the PEST_HP manager command line window. If all went well, you should see text that is similar to the following in the window:

```
Waiting for at least one agent to appear....

    New agent has appeared:-
    Agent host: "10.24.97.28"
    Agent working directory: "..\_tutorial\agent_dir\agent0\pest"
    Number assigned to agent: 1
    Agent speed index:   106.67

    - number of runs completed...
          1
    Sum of squared weighted residuals (ie phi)              =   3821.8
    Contribution to phi from observation group "heads1"     =   334.09
    Contribution to phi from observation group "heads3"     =   3401.8
    Contribution to phi from observation group "streams"    =   47.618
    Contribution to phi from observation group "headiff"    =   38.304
    Contribution to phi from observation group "pstreams"   =   0.0000

    Optimisation complete: optimisation iteration limit of 0 realized.
    Total model calls:      1

 See file calib1.rec for full run details.
 See file calib1.ofr for objective function record.
 See file calib1.sen for parameter sensitivities.
 See file calib1.seo for observation sensitivities.
 See file calib1.res for residuals.
 See file calib1.svd for history of SVD process.
 See file calib1.rmf for run management history.
 See file calib1.rme for run management efficiency.
 See file calib1.par for best parameters.
```

51 As you can see, PEST_HP has run the model once, and then recorded the contribution made by each observation group to the total objective function. The contribution from group *heads3* is at least an order of magnitude greater than that of any of the other groups; it is two orders of magnitude larger than that of the stream flow observation group (*streams*).

52 If you list the contents of the PEST_HP managers' folder (i.e. the *pest* folder), you will see a set of new PEST output files. Feel free to inspect these files with a text editor if you wish. See the PEST manual for a detailed description of what each file contains. Some of these will be discussed later in the tutorial.

Great, the PEST setup is functional, and everything seems to work. We can now proceed to fine tune PEST setup.

4.2 Weighting for Visibility

Prior to estimating parameters using PEST, a user must decide how to weight observations. In some cases, it is wise to weight observations strictly according to the inverse of the standard deviation of measurement noise. Certainly, observations of greater credibility should be given greater weights than those of lower credibility. However, in many history-matching contexts, model defects are just as instrumental in inducing model-to-measurement misfit as is the noise associated with field measurements. Model imperfections affect some types of model outputs more than others. Also, their effects on model output differences (spatial or temporal) are often less than on raw model outputs.

It follows that a history-matching process can often benefit from the inclusion of measurement differences and corresponding model output differences in a calibration dataset, It can also be improved by dividing components of a measurement dataset into groups based on type and location, assigning these measurements to different observation groups, and by ensuring that the contributions made by each of these groups to the original objective function are about the same. The information content of each of these groups is thereby given equal right of entry to the parameter estimation process. This matter is extensively discussed in the PEST Book.

As we saw in step 26, the current weighting scheme results in objective function contributions from different observation groups that range over two orders of magnitude. In particular, the contribution from the *streams* observation group is comparatively small. Given that our prediction of interest is groundwater discharge to the stream, intuition suggests that the weight ascribed to this observation group should be increased. However, we also want to ensure that the information content of other observation groups is not ignored.

As stated above, a practical means of accommodating this situation is to weight all observation groups such that they contribute an equal amount to the starting measurement objective function. In this manner, no single group dominates the objective function, or is dominated by others; the information contained in each group is therefore equally "visible" to PEST.

The PEST PWTADJ1 utility automates this process of "weighting for visibility". PWTADJ1 is described in Part 2 of the PEST manual. PWTADJ1 reads an existing PEST control file (**.pst*) and a corresponding run record file (**.rec*). This means that PEST must have already been run at least once – preferably with NOPTMAX set to 0 or 1. Recall that we did this in Chapter 4.1).

53 Open a command line window in the *tutorial\pest* folder (the PEST_HP manager folder), type the following command and then press <enter>:

```
pwtadj1 calib1.pst calib1-wt.pst 1000
```

54 The following should appear on the screen:

```
- reading PEST control file calib1.pst for first time...
- file calib1.pst read ok.

- reading PEST run record file calib1.rec...
- file calib1.rec read ok.

- re-reading file calib1.pst and writing file calib1-wt.pst...
- file calib1.pst read ok.
- file calib1-wt.pst written ok.

Warning: an adjustment factor cannot not be computed for members of
obsevation group "pstreams" as contribution to objective function of this
group is zero.
```

55 The above command instructs PWTADJ1 to create a new PEST control file named *calib1-wt.pst* based on an existing PEST control file named *calib1.pst* and a complementary run record file

named *calib1.rec*. PWTADJ1 reads contributions made by different observation groups to the objective function from the latter file. It then adjusts multiplies existing observation weights with a group-specific weight factor such that each group contributes *1000* to the overall objective function in the new PEST control file. In doing this, relativity of observation weighting within each group is preserved.

56 Inspect the folder. You should see a new PEST control file named *calib1-wt.pst*. Open it in a text editor and inspect the "observation data" section. Compare the observation weights in this new file to those in *calib1.pst*. Note that each observation group now has altered (and distinct) weights.

57 Let us see what this has done to the overall contribution to the objective function. As we did in chapter 4.1, run PEST_HP with the new *calib1-wt.pst* PEST control file with NOPTMAX set to zero in that file.

58 Repeat steps 39 to 48 for *calib1-wt.pst*. Replace *calib1.pst* with *calib1-wt.pst* when starting the PEST_HP agent and manager.

59 The command to start the agent becomes:

```
agent_hp calib1-wt.pst /h %computername%:4004
```

60 The command to start the manager becomes:

```
pest_hp_mkl calib1-wt.pst /h :4004
```

61 Once complete (it should take about 1 second to run), you should see the following in the manager's command line window:

```
- number of runs completed...
     1
Sum of squared weighted residuals (ie phi)             =   4000.0
Contribution to phi from observation group "heads1"     =   999.99
Contribution to phi from observation group "heads3"     =   999.99
Contribution to phi from observation group "streams"    =   1000.0
Contribution to phi from observation group "headiff"    =   999.99
Contribution to phi from observation group "pstreams"   =   0.0000

Optimisation complete: optimisation iteration limit of 0 realized.
Total model calls:      1
```

62 As you can see, with the exception of *pstreams,* contributions to the objective function are now all (close to) 1000.0, and therefore close to equal. The observation group *pstreams* is an observation of a "prediction". It does not contribute to the history-matching process. That is why it was originally assigned a weight of zero when constructing the first PEST control file (see the OLPROC script *olproc-pred.in* in Chapter 3.4). As observations belonging to the *pstreams* group in the existing PEST control file (*calib1.pst*) all had weights zero, PWTADJ1-adjusted weights for these observations in the new control file (*calib1-wt.pst*) will also be zero.

All files produced up to this point can be found in the *completed\pest-calib-1-wt* folder provided in the tutorial directory. These form the starting point for the next chapter.

4.3 Adding Tikhonov Regularization

The mathematical term for the process through which a unique solution is sought for a nonunique inverse problem is "regularisation". The goal of regularised inversion is to seek a unique parameter field that results in a suitable fit between model outputs and field measurements, whilst minimizing the potential for wrongness in model predictions. That is, out of all the ways to fit a calibration dataset, regularized inversion seeks the parameter set of minimum error variance.

One way to seek a parameter field of minimum error variance is to seek a parameter field that allows the model to fit the calibration dataset, but whose values are also as close as possible to a set of "preferred parameter values". Ideally, preferred parameter values should also be initial parameter values as listed in the "parameter data" section of the PEST control file. These preferred parameter values are normally close to the centre of the prior parameter probability distribution. At the same time, scales and patterns of departures from the preferred parameter set that are required for model outputs to fit a calibration dataset should be achieved in ways that make "geological sense".

PEST provides a user with a great deal of flexibility in how Tikhonov constraints can be introduced to an inversion process. The easiest way is to do this is through the use of prior information equations. When prior information equations are employed, Tikhonov constraints are expressed through preferred values that are assigned to linear relationships between parameters. In our case preferred values will be applied to the parameters themselves. (Equality is the simplest type of linear relationship.) Weights must be assigned to these equations. As is described in PEST documentation, when PEST is run in "regularisation" mode, it makes internal adjustments to the weights that are assigned to any observations or prior information equations that belong to special observation groups that are referred to as "regularisation groups".

As for non-regularisation observations and prior information equations, weights can be replaced by covariance matrices. These are rarely used with normal observations. However when observations or prior information equations embody Tikhonov constraints, they can be used to ensure that patterns of parameter heterogeneity that emerge from the inversion process are geologically reasonable.

Covariance matrices are preferred over weights for prior information equations that embody Tikhonov constraints. However, we will demonstrate both options in the following workflow.

Prior Information Equations

The ADDREG1 and ADDREG2 utilities (see Part 2 of the PEST Manual) automate the addition of prior information equations to a PEST control file. These utilities also add a "regularisation" section to the control file. ADDREG1 sets a target measurement objective function (PHIMLIM) which is very low (1.0E-10), whilst ADDREG2 sets it to a user-specified value.

In practice, it is often convenient to commence the model calibration process by assigning a very low value to PHIMLIM. At the same time, the FRACPHIM regularisation control variable should be set to 0.1 to ensure that Tikhonov constraints are still active. With the target measurement objective set very low, it is up to you, the user, to halt PEST once it is apparent that it has reduced the measurement objective function to a reasonable value. At this point, PEST-estimated parameter value can either be accepted, or PEST can be re-run with PHIMLIM set to a value that prevents too good a fit from incurring unrealistic parameter values. (The first PEST run will have informed you when fits are too good; that is when the measurement objective function is too low.)

It is worth noting that in an ideal inversion context in which observation weights are inversely proportional to the standard deviation of measurement noise, a PHIMLIM setting equal to the number of non-zero-weighted observations provides a defensible trade-off between attaining an acceptable level of model-to-measurement fit and adherence to the preferred parameter condition. However, as was discussed above, this "ideal" situation rarely exists because model defects and imperfections generally make a substantial contribution to model-to-measurement misfit.

For the current tutorial we will use ADDREG1 and simply set PHIMLIM very low.

63 Open the command line in the *tutorial\pest* folder, type the following command and then press <enter>:

```
addreg1 calib1-wt.pst calib1-wt-reg.pst
```

64 You should see the following text on your screen:

```
- file calib1-wt.pst read ok.
- file calib1-wt-reg.pst written ok.
```

65 In the folder you should see a new file named *calib1-wt-reg.pst*. Open it with a text editor and inspect it.

66 Note that the “control data” section of the PEST control file has been updated. Values for the **NPRIOR** and the **NOBSGP** variables have been changed.

```
* control data                       NPRIOR
restart regularisation
2205      23        7     2205      11 <-- NOBSGP
3  5    single  point
10.0  -3.0  0.3  0.03  10  lamforgive
30.0  30.0  0.001
0.1
0  0.005  4  4  0.005  4
0  0  0
```

67 If you inspect the “observation groups” section of the PEST control file, you will see six new observation groups. Their names begin with “*regul_*”; this is followed by the name of a parameter group.

68 If you scroll to the end of the PEST control file, you will find a new “prior information” section. This section contains 2205 equations which specify preferred values for all parameters. The preferred value assigned to each parameter is its initial value as specified in the “parameter data” section of the PEST control file.

It is worthy of note that, as stated above, Tikhonov regularisation can be implemented in ways other than through the assignment of preferred values to parameters. Indeed, utilities available through the PEST Groundwater Utility suite provide other options, including “preferred difference” regularisation. Alternatively, you can write your own equations (a task that is best automated to reduce the chances of error) or provide nonlinear regularisation constraints through appropriate submodels for which the PEST control file provides pertinent “observed values”. However, in this tutorial we will employ the simplest option. In many real-world circumstances it is also the most effective option.

69 As usual after creating a new PEST control file, it is a good idea to run PESTCHEK to verify its integrity.

Covariance Matrices

The use of prior information equations to specify preferred values for parameters should ideally be accompanied by the assignment of covariance matrices to these equations. In principle, the use of covariance matrices ensures that if the values of calibrated parameters must depart from the preferred values for these parameters, then they do so in “reasonable ways”. For pilot point parameters a covariance matrix should be associated with each group of prior information equations; generally a group of prior information equations specifies preferred values for members of a particular parameter group. (This strategy assumes that pilot point parameters are divided into groups according to parameter type, layer, and maybe zone.) Generally the use of a covariance matrix encourages PEST to spread emerging parameter heterogeneity over a group of neighbouring pilot points rather than attributing it to a single pilot point. This reduces the propensity for the appearance of “bulls-eyes” in a calibrated parameter field. This results in a smooth parameter field that, by virtue of its smoothness, is more likely to be of minimized error variance. That is, it is central with respect to a collection of more “detailed” parameter fields that can also fit the calibration dataset, but for which such detail cannot be inferred uniquely. Carefully-constructed Tikhonov regularisation can also introduce expert knowledge on the nature, disposition and anisotropy associated with geological structures that may be reflected in calibrated parameter fields.

Covariance matrices for each of the pilot point parameter groups have been previously prepared. They are provided for you in the *runmodel\preproc* folder. These are the files with the **.mat* extension. (See the GMDSI tutorial on the PPCOV* suite of utility programs and [PEST Roadmap 7: A Pilot Points Workflow]() for details on how to prepare covariance matrices.

These covariance matrices need to be attributed to the respective regularisation group.

70 Make a copy of the *calib1-wt-reg.pst* PEST control file. Name the copy *calib1-wt-cov.pst*. Open the new control file in a text editor.

71 In steps 63 to 69, ADDREG1 added regularisation groups for each parameter group to the PEST control file. If you inspect the "observation group" section of *calib1-wt-cov.pst*, you will see seven new groups, all with the prefix "*regul_*". These are the regularisation groups.

```
* observation groups
heads1
heads3
streams
headiff
pstreams
regul_kx1
regul_kx2
regul_kx3
regul_rch
regul_strc
regul_ghb1
regul_ghb3
```

72 To assign each respective covariance matrix to its correct regularisation group, record the name of the covariance matrix alongside the regularisation group name. If this matrix does not reside in the PEST_HP manager working folder, then include its path with the name. For example:

```
* observation groups
heads1
heads3
streams
headiff
pstreams
regul_kx1     ..\runmodel\preproc\cov1.mat
regul_kx2     ..\runmodel\preproc\cov2.mat
regul_kx3     ..\runmodel\preproc\cov3.mat
regul_rch     ..\runmodel\preproc\cov_rch.mat
regul_strc    ..\runmodel\preproc\cov_stream.mat
regul_ghb1    ..\runmodel\preproc\cov_ghb.mat
regul_ghb3    ..\runmodel\preproc\cov_ghb.mat
```

73 Alternatively, you can automate this process using the ADDCOVMAT utility (see Part 2 of the PEST manual). This can be advantageous if you need to repeat this process many times.

74 ADDCOVMAT requires as input an existing PEST control file and a text file linking names of covariance matrix files to observation groups.

75 Create a new text file in the *pest* folder. Name it *covmatfile.in*. Type the following text into the file and then save it:

```
regul_kx1       ..\runmodel\preproc\cov1.mat
regul_kx2       ..\runmodel\preproc\cov2.mat
regul_kx3       ..\runmodel\preproc\cov3.mat
regul_rch       ..\runmodel\preproc\cov_rch.mat
regul_strc      ..\runmodel\preproc\cov_stream.mat
regul_ghb1      ..\runmodel\preproc\cov_ghb.mat
regul_ghb3      ..\runmodel\preproc\cov_ghb.mat
```

76 Open a command line window in the *pest* folder, type the following command and then press <enter>:

```
addcovmat calib1-wt-reg.pst covmatfile.in calib1-wt-cov.pst
```

77 You should see the following text appear on the screen:

```
- 7 observation groups featured in file
- file calib-wt-reg.pst read ok.
- file calib-wt-cov.pst written ok.
```

78 Inspect the *pest* folder. You should see a new *calib1-wt-cov.pst* file. Inspect the "observation group" section of this control file. You should see that the names of the covariance matrices have been added to this file. With this control file, PEST will ignore the weights associated with prior information equations. Instead of weights, it will use the observation-group-specific covariance matrices when computing the contribution that these prior information equations make to the overall objective function.

79 As usual after creating a new PEST control file, it is a good idea to run PESTCHEK to verify its integrity. This will also alert you to potential issues with the covariance matrices.

At this stage we have two version of the PEST control file ready to start calibration. Both are weighted for visibility of all observation groups. The first PEST control file, *calib1-wt-reg.pst*, includes Tikhonov regularisation with preferred values. The second PEST control, *calib1-wt-cov.pst*, also assigns preferred values to adjustable parameters. However, instead of ascribing weights to the prior information equations which provide preferred parameter values, it ascribes covariance matrices to these equations In both cases, in accordance with its standard regularisation functionality, PEST calculates a "regularisation weight factor" that it applies to all observations and prior information equations that belong to groups whose names begin with "*regul*". This iteration-specific weight factor is designed to ensure that the measurement objective function achieved through the current iteration of the inversion process is equal to the target measurement objective function (PHIMLIM), or FRACPHIM times the current value of the measurement objective function, whichever is greater. Because this generally cannot be achieved during a single iteration, another iteration generally follows the current iteration. See PEST documentation for full details.

4.4 Some Final Tweaks

When using PEST_HP it is recommended that the UPTESTMIN variable be set to 20 or higher. This ensures that PEST_HP carries out at least 20 model runs when testing parameter upgrades. If PEST_HP has more than 20 CPU cores at its disposal, it will automatically use all of them to carry out more lambda-testing model runs.

It is sometimes convenient to inform PEST that it should record parameters that it estimates, and model-to-measurement residuals that it achieves, at the end of every iteration of the inversion process. PEST's default behaviour is to update these during every iteration, and to record only the updated values, thereby over-writing values achieved during previous iterations. This strategy allows a user to return to access iteration-specific outcomes of the inversion process. It is particularly useful in cases where PHIMLIM has been set very low. As was mentioned previously, PEST will probably not be able to achieve this value. However in trying to do so it will lower the measurement objective function as far as it can. This will almost certainly result in “distortion” of parameter fields as “over-fitting” starts to occur. If iteration-specific inversion results are saved, a user can retrieve parameter values obtained at the end of the iteration at which the objective function started to flatten out, but at which the parameter field was more reasonable. Adding the PARSAVEITN and REISAVEITN variables to the “control data” section of a control file activates this behaviour.

80 Update the “control data” section of both control files as shown below:

```
* control data
restart estimation
2205      23       7       0       5                  UPTESTMIN
3  5    single  point
10.0  -3.0  0.3  0.03  10  lamforgive uptestmin=20
30.0  30.0  0.001
0.1                              PAR & REISAVEITN
0  0.005  4   4 0.005  4
0  0   0 parsaveitn reisaveitn
```

The PEST two control files *calib1-wt-reg.pst* produced in this chapter can be found in the *completed\pest-calib-2-reg* folder provided in the tutorial directory. The *calib1-wt-cov.pst* file can be found in the *completed\pest-calib-3-cov* folder. Either of these files can be used as a starting point for Chapter 5

5. CALIBRATION

The model runs relatively fast; it has a run time of 1 to 2 seconds. Nevertheless, because of the number of parameters that are featured in the inversion process, this process can still take some time. How long it takes depends on available computing resources. For example, on an i9 Intel CPU with 10 cores running 10 agents, calibration required between 40min and 2.5h. (Now take a moment, dear reader, and ponder the implications for cases in which single model runs take minutes or hours...).

To save time, both variants of the calibration process have been run for you. PEST output files are provided in folders named *completed\pest-calib-2-reg* and *completed\pest-calib-3-cov*. The respective model run files with best-fit parameters are provided in *completed\runmodel-calib-reg* and *completed\runmodel-calib-cov,* respectively.

However, if you wish to undertake the calibration process yourself, carry out the following steps.

5.1 Starting the Calibration

These steps are described only for the *calib1-wt-**reg**.pst* case. To calibrate the model using the *calib1-wt-**cov**.pst* PEST control file, simply repeat these steps, replacing the name of the PEST control file where necessary.

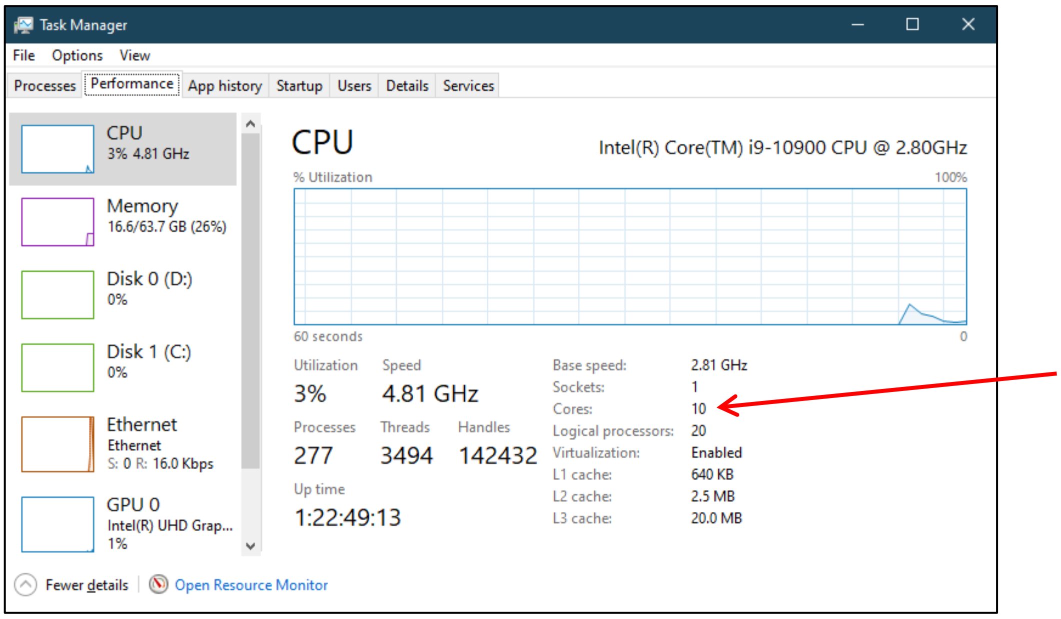
81 First, change NOPTMAX to 50.

```
* control data
restart estimation
2205      23      7      0      5
3  5    single  point
10.0  -3.0  0.3  0.03  10  lamforgive uptestmin=20
30.0  30.0  0.001
0.1
50  0.005  4  4  0.005  4
0  0  0 parsaveitn
```

NOPTMAX

Next, prepare all the agent folders and start up the PEST_HP agent processes. To speed up the calibration process, you will need to distribute the workload across as many parallel agents as possible. Normally, you will want to use the same number of agents (or less) as you have available CPU cores. Most personal computers (i.e. desktops or laptops) these days have between 4 and 10 cores. Servers or HPCs may have many more cores than this.

82 If you are on a Windows operating system, to find out how many cores you have press *Ctrl+Shift+Esc*. This will open your Task Manager window. Click on the Performance tab. Here you will see the number of available cores. For example, the machine pictured in the screenshot below has 10 cores.


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
CPU 3% 4.81 GHz
Memory 16.6/63.7 GB (26%)
Disk 0 (D:) 0%
Disk 1 (C:) 0%
Ethernet Ethernet S: 0 R: 16.0 Kbps
GPU 0 Intel(R) UHD Grap... 1%
CPU
Intel(R) Core(TM) i9-10900 CPU @ 2.80GHz
% Utilization
100%
60 seconds
0
Fewer details
Open Resource Monitor

Utilization 3%
Speed 4.81 GHz
Processes 277
Threads 3494
Handles 142432
Up time 1:22:49:13

Base speed:	2.81 GHz
Sockets:	1
Cores:	10
Logical processors:	20
Virtualization:	Enabled
L1 cache:	640 KB
L2 cache:	2.5 MB
L3 cache:	20.0 MB

83 In your *agent_dir* folder, create as many "agent" folders as you wish to use. For example, if you have four cores, create four folders named *agent1, agent2, agent3* and *agent4*. Repeat steps 46 to 48 for each of those folders. (As you can see, being able to do this programmatically will make your life a lot easier – especially when dealing with many agents. For example, it is possible to automate this process using a batch file or a simple Python script.)

84 At this point you should have several command line windows open, each with an agent waiting for instructions from the PEST_HP manager.

85 Open a command line window in the *tutorial\pest* folder and run the PEST_HP manager with *calib1-wt-reg.pst* by providing the following command:

```
pest_hp_mkl calib1-wt-reg.pst /h :4004
```

At this point you get that moment of feeling badass as all the command line windows kick-off and start reporting model outputs to their individual screens. Once the feeling has passed and you have returned to reality, it might be worth finding something else to do for a while.

If you allow it to run to completion, the calibration will likely take a while. That being said, you rarely need to let it run all the way to the end. After 6 iterations the fit is near perfect (the same occurs for *calib1-wt-cov.pst*) and the calibration process can be halted if you wish. (The six iterations took about 45 minutes on an i9 CPU with 10 agents). If you allow it to keep going, it will reach 13 iterations; this took approximately 2 h with an i9 CPU using 10 agents.

86 To stop the calibration process simply go to the PEST_HP manager command line window and press *Ctrl+C* on your keyboard.

87 Alternatively, open a new command line window in the same from folder in which the PEST_HP manager is running, and type in the following command and then press <enter>.

```
pstopst
```

The next chapter describes how to monitor the calibration progress and why you may choose to halt it before completion.

5.2 Monitoring the Calibration

Whilst PEST_HP is running, it updates several output files with information that can be of use for monitoring the calibration process. It also writes some summary outputs to the screen. It can be useful to keep an eye on these outputs while PEST is running. Sometimes you might catch a mistake or odd behaviour, allowing you to correct or adapt the setup before losing too much time.

88 If you inspect the *pest* folder whilst PEST is running (or the provided folder with the outcome of the calibration process), you will see a series of files with the same filename base as the PEST control file, but with a different extension for each. These are PEST output files. (See PEST documentation for detailed descriptions of each of these file types.)

89 The "objective function record" file (**.ofr*) contains a record of the objective function, together with contributions to the total objective function from each observation group; values of all of these are provided for each iteration. This information can be useful to get a "big picture" of how PEST is doing, and which observation groups are giving it the most trouble.

90 Open *calib1-wt-reg.ofr* in a text editor. (You can do so whilst PEST is running, as long as the text editor software you are using does not lock the file. Most editors, such as Notepad++, TextEditor, PSPad and so on, allow this.). The first few lines should look something like this (the font has been reduced so that it fits on the page):

```
iteration    measurement    regularisation      heads1     heads3     streams        headiff        pstreams
    0      3999.97       2.883599E-11    999.987       999.988       1000.01        999.989        0.00000
    1      2698.85       0.160978        843.823       932.168       922.737       0.123820        0.00000
    2      643.123        8.81611        186.751       362.363       85.3471        8.66164        0.00000
    3      26.4909        33.7586        7.42489       11.5622       2.19411        5.30967        0.00000
    4      2.98643        32.7271       0.114301       2.67006      0.152262      4.980816E-02     0.00000
    5     0.338608        34.3971      4.059894E-02   0.206422      1.485518E-02   7.673226E-02     0.00000
(…)
```

91 During each iteration of the inversion process, PEST updates this file. As you can see, PEST did not have much trouble here; it managed to drop the objective function for all groups within a few iterations. Figure 4 shows a plot of values recorded in *calib1-wt-reg.ofr*. As you can see, by the fifth iteration PEST has already dropped the objective function by several orders of magnitude. (Note that the y-axis shows log-transformed objective function values). However, as we had set a

very low value for the PHIMLIM control variable (i.e. the target measurement objective function), PEST continues searching for a near perfect fit.

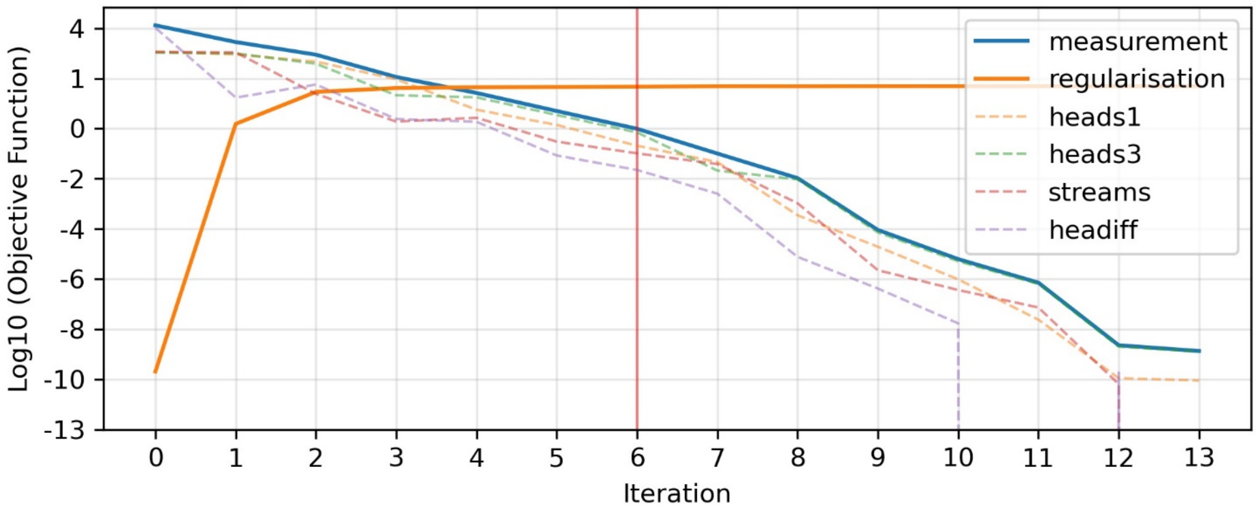


Figure 4 - Evolution of objective function components at each iteration during the *calib1-wt-res.pst* calibration. The vertical red line marks iteration 6.

92 The residuals (i.e. differences) between simulated and measured observations at the end of each iteration of the inversion process can be found in the *.*rei* files. By default, PEST only records residuals from the latest iteration in the *.*rei* file, thereby overwriting results from previous iterations. However, as we used the REISAVEITN control variable, PEST records a *.*rei* at the end of each iteration. These are files named *calib1-wt-res.rei.1, calib1-wt-res.rei.2,* and so on. By inspecting these files, we are able to review the evolution of residuals as they change at each observation point at each iteration. This gives us an additional way in which to assess at what point model-to-measurement fit is “good enough”, and to thereby consider halting PEST execution.

93 Open a few of the *.*rei* files in a text editor and inspect them. If you open the file for iteration 6 (*calib1-wt-res.rei.6*), you should see the following:

```
MODEL OUTPUTS AT END OF OPTIMISATION ITERATION NO.   6:-
Note that weights have been adjusted in accordance with regularisation target objective function.

 Name                 Group          Measured         Modelled         Residual         Weight
 h1-3892_1            heads1         89.84313         89.84238         7.5410000E-04    17.30090
 h1-1664_1            heads1         84.66009         84.66604        -5.9476000E-03    17.30090
 h1-162_1             heads1         85.48545         85.50325        -1.7801200E-02    17.30090
 h3-3521_1            heads3         94.85694         94.87348        -1.6544900E-02    5.421830
 h3-3988_1            heads3         94.84176         94.84738        -5.6232000E-03    5.421830
 h3-3417_1            heads3         91.62516         91.62525        -9.0300000E-05    5.421830
 h3-3829_1            heads3         90.06320         90.10091        -3.7711900E-02    5.421830
 h3-3357_1            heads3         88.31318         88.35669        -4.3507500E-02    5.421830
 h3-3594_1            heads3         89.08431         89.08621        -1.9016000E-03    5.421830
 h3-3892_1            heads3         89.73940         89.73845         9.4630000E-04    5.421830
 h3-1152_1            heads3         84.79649         84.84354        -4.7053600E-02    5.421830
 h3-313_1             heads3         98.44412         98.43816         5.9588000E-03    5.421830
 h3-209_1             heads3         90.80855         90.82414        -1.5590000E-02    5.421830
 h3-331_1             heads3         88.26986         88.30780        -3.7944900E-02    5.421830
 h3-707_1             heads3         86.52537         86.54495        -1.9576900E-02    5.421830
 h3-3956_1            heads3         87.48643         87.47381         1.2620600E-02    5.421830
 h3-2270_1            heads3         84.60294         84.67374        -7.0804400E-02    5.421830
 h3-521_1             heads3         85.67934         85.67713         2.2128000E-03    5.421830
 trib_1               streams       -8385.727        -8388.056         2.329350        5.4647990E-03
 total_1              streams      -11233.40        -11288.30         54.90210        4.0794680E-03
 dh3892_1             headiff       0.1037272        0.1039222       -1.9496000E-04    510.9490
```

94 The fit in iteration 6 is already remarkably good. (A bit too good! Recall our “assumed” measurement noise in chapter 2.4.) The residuals are already less than could be explained by measurement error.

95 Figure 5 summarizes the residuals per observation group at each iteration. (The figure was generated using a Python script; similar plots could be achieved with other software such as Grapher or MSExcel.) Each box-and-whiskers plot represents the statistical distribution of residuals at each iteration for a given observation group. For example, looking at the top two plots for observation groups *heads1* (top left) and *heads3* (top right), even by the fourth iteration most residuals are less than can be accounted for by measurement error (i.e. less than +/-0.1 m). The residuals for observations in the *streams* group are also at an acceptable value after the third iteration. Observations in the *headiff* group are near a perfect fit even at the first iteration. You can confirm this by inspecting the respective **.rei* files.

96 Had we been monitoring the PEST run, we might have halted it after completing the sixth iteration, saving some computation time.

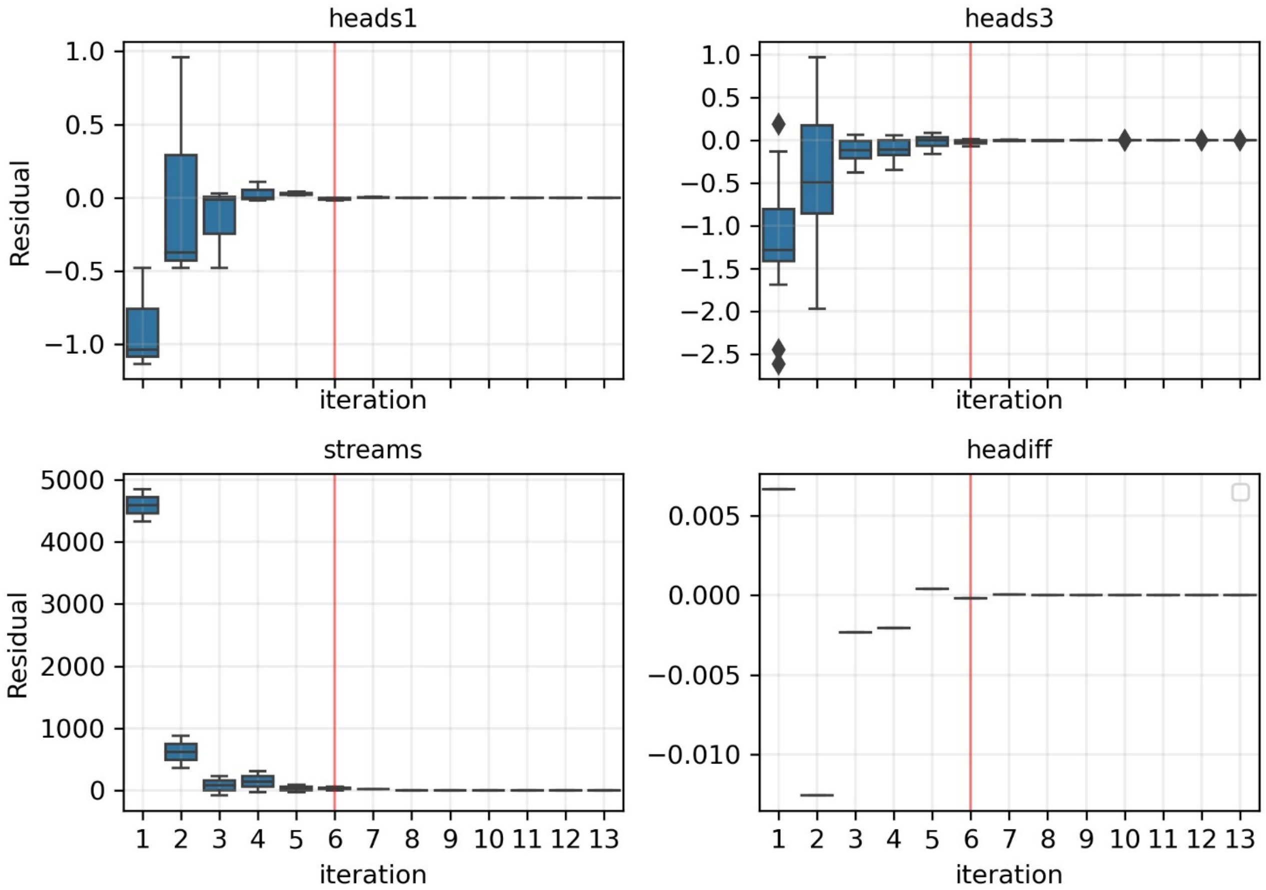


Figure 5 - Boxplots of residuals between simulated and measured observation at each iteration for each observation group obtained from the *calib1-wt-reg.pst* calibration run. The vertical red line highlights the outcomes at iteration 4.

5.3 Calibrated Parameters

At any stage of an inversion process, best parameters achieved to date are recorded in a PEST “parameter value file”. This file is named *casename.par*; *casename* is the filename base of the PEST control file. For example, the PEST run based on the *calib1-wt-reg.pst* PEST control file results in a parameter value file named *calib-wt-reg.par*. If you inspect the *pest* folder either during or after the calibration process (or the folders with outcomes from calibration runs provided for you in the *completed* folder), you should see such a file. If you inspect it with a text editor, you will see a list of parameter names and corresponding parameter values. These correspond to the best fit that PEST was able to achieve.

To display these parameters spatially, they need to be assigned to model input files. To transfer these parameter values to model input files, they need to be assigned as initial parameters values in a new PEST control file. This can be accomplished using the PARREP utility. Then, the new PEST control file can be used to assign hydraulic property values based on optimized parameters to model input files and run the model once (by setting NOPTMAX equal to zero), as will be demonstrated below.

However, as has already been discussed, if you let PEST run for many iterations, the measurement objective function may fall only slightly during later iterations of an inversion process while parameter fields may become progressively more unreasonable as PEST strives to squeeze out the last bit of improved fit. In such cases, it can be useful to return to a parameter set from an earlier iteration, thereby accepting a slightly worse fit in exchange for a more "reasonable" parameter field. By using the PARSAVEITN variable in the "control data" section of a PEST control file, a user can record the best parameters obtained at each iteration and thus return to the iteration of their choice. In the *pest* folder (the same folder in which we ran the PEST_HP manager or, if you chose not to carry out calibration yourself, in the *completed\pest-calib1-reg* folder), these are the files named *calib-wt-reg.par.1*, *calib-wt-reg.par.2*, etc.. These files house optimized parameters obtained in iteration 1, 2, and so on.

For the purposes of the tutorial, let us assume that for the PEST run based on file *calib1-wt-reg.pst* we wish to accept the parameter field obtained during iteration 6. These are the parameters housed in the file named *calib1-wt-reg.par.6*.

97 Open a command line window in the *pest* folder; run PARREP by typing the following command and pressing <enter>:

```
parrep calib1-wt-reg.par.6 calib1-wt-reg.pst reg-par6.pst 0
```

98 You should see the following text appear on the screen:

```
Reading parameter value file calib1-wt-reg.par.6 ----->
Data for 2205 parameters read from file calib1-wt-reg.par.6.

Reading file calib1-wt-reg.pst and writing file reg-par6.pst ----->
File reg-par6.pst written ok.
```

99 The above command instructs PARREP to write a new PEST control file named *reg-par6.pst*, based on *calib1-wt-reg.pst*, but with the initial parameter values obtained from the *calib1-wt-reg.par.6* file. It also sets NOPTMAX in the new *par6-reg.pst* to zero.

100 Proceed to run PEST_HP with *reg-par6.pst* (use a single agent). The model will run once. Once PEST_HP has finished, the model input and output files in the agent folder pertain to the "calibrated" model.

101 Make a copy of the *runmodel* folder of the "calibrated" model. Name the folder *runmodel-calib-reg* and place it in the tutorial working folder for safekeeping. (That being said, as long as you have the *par6-reg.pst* PEST control file, you can always recreate this model run by repeating the above steps.)

The model input and output files can now be used to visualise parameter fields and simulation outcomes. In the following chapter we will take a look at the hydraulic conductivity (K) distribution emerging from the two regularisation schemes (i.e. with/without assignment of covariance matrices to prior information equations). We will also compare them with the "real" K distribution.

5.4 Comparing Calibration Outcomes

Figure 6 and Figure 7 show the calibrated hydraulic conductivity fields obtained with preferred value regularisation accompanied, and unaccompanied, by the use of covariance matrices respectively. (See GMDSI tutorials on model visualisation; alternatively, you can achieve similar visualisations using a GUI or FloPy.)

Both calibration processes achieved comparable levels of (almost perfect) fit with measurements comprising the calibration dataset. Note, however, that the spatial distribution of hydraulic conductivity exhibited by Figure 6 is notably “blotchy” (in particular in layer 3); on the other hand, in Figure 7 heterogeneity is spread out so that areas of high or low K are represented as smoother, larger-scale structures. Where prior information equations that express preferred parameter values are unaccompanied by covariance matrices (Figure 6), individual pilot points depart from their initial values on a point by point basis, and hence only locally, where this improves the fit between model outputs and field measurements. Elsewhere, the initial value is retained. Where covariance matrices are provided with prior information equations (Figure 7), this induces the parameter estimation process to spread homogeneity among neighbouring pilot points if it can – for the objective function penalty resulting from distributed heterogeneity is less than that resulting from point-by-point heterogeneity.

Unfortunately, it is the nature of model calibration, that while one of the parameter fields is more aesthetically pleasing, and can lay greater claim to a status of “minimized error variance” (namely the parameter field of Figure 7), both are wrong.

Figure 7 shows the hydraulic conductivity field used to simulate “reality”. Compare this with Figure 6 and with Figure 7. As you can see, PEST was able to achieve an excellent fit with the calibration dataset with a fraction of the heterogeneity that exists in “reality”. Some of the patterns visible in Figure 7 are reflected in the calibrated parameter fields; there are some similarities between patterns of heterogeneity exhibited by Figure 7 and Figure 7. However neither of the calibrated parameter fields captures the heterogeneity (or even the range of parameter values) of “reality”. This is because the information content of the calibration dataset is insufficient to allow this. The calibration process therefore yield the minimum amount of heterogeneity that is required to fit the calibration dataset; Thanks to the use of covariance matrices, the patterns of heterogeneity are more reasonable in Figure 7 than they are in Figure 6.

What Figure 6 and Figure 7 show is the spatial heterogeneity that is informed by the calibration dataset, represented according to the “soft” knowledge embedded in the Tikhonov regularisation scheme. Where that “soft knowledge” includes some guidance on the nature of the patterns of heterogeneity that should emerge when fitting a calibration dataset (embodied in the present case in covariance matrices), the parameter fields that emerge from the inversion process are “better”. Though not discussed further herein, “better” means that model predictions made by the calibrated model are more likely to lie in the centres of their posterior probability distributions. If these probability distributions are broad, then these predictions may be considerably in error, notwithstanding the calibrated status of the model.

Other tutorials will show you how to explore the post-calibration uncertainties of parameters and predictions. Both linear and nonlinear methods will be explored. Linear methods are easy to apply and can yield a plethora of supplementary information. Nonlinear methods are more difficult to apply but support a more accurate characterization of post-calibration parameter and predictive uncertainty.

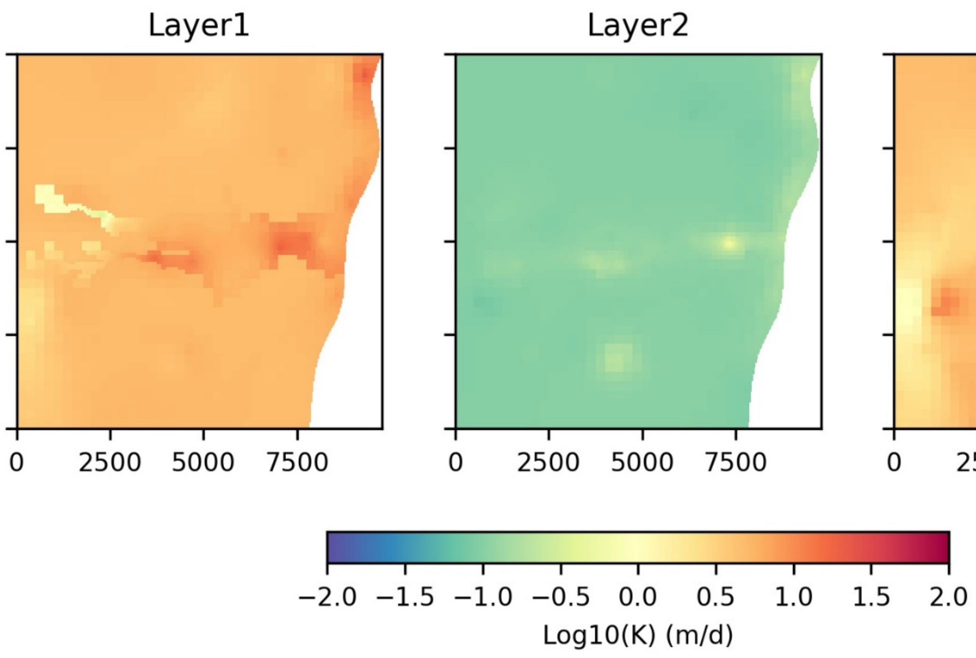


Figure 6 - Spatial distribution of K (coloured using log10 scale) in each layer for from iteration 6 with Tikhonov regularisation implemeted using prior information equations expressing preferred parameter values (*calib1-wt-reg.pst*). Individual weights are ascribed to these prior information equations.

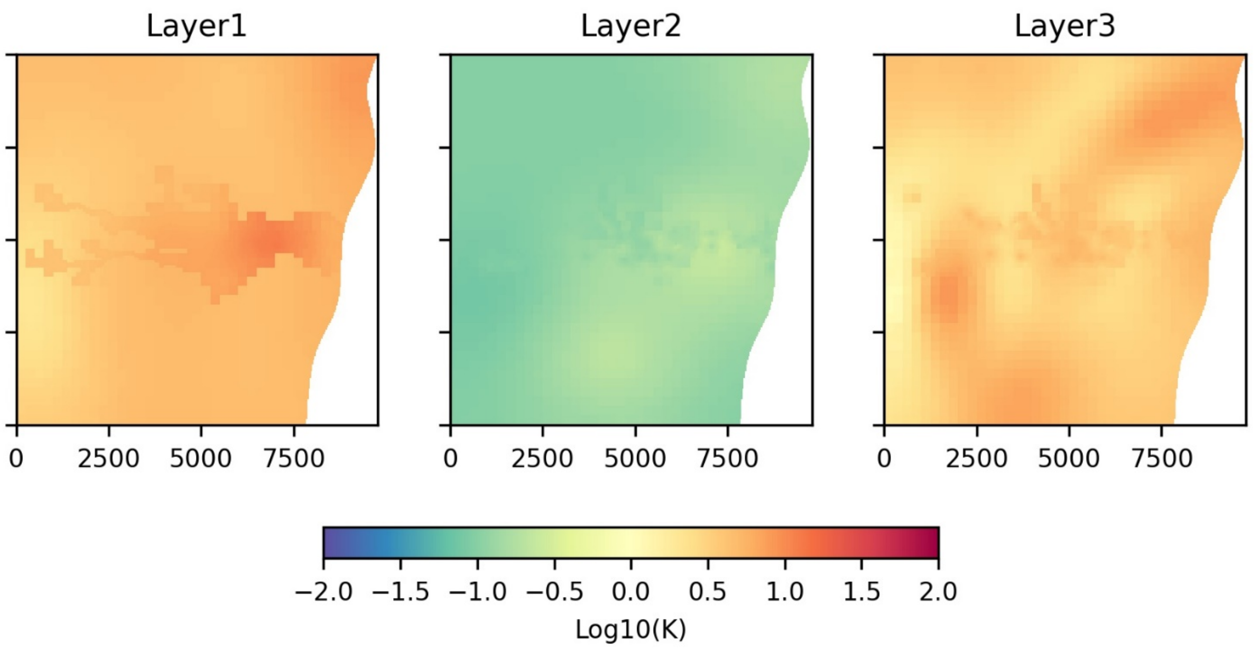


Figure 7 - Spatial distribution of K (coloured using log10 scale) in each layer for from iteration 6 with Tikhonov regularisation implemeted using prior information equations expressing preferred parameter values (calib1-wt-cov.pst). Covariance matrices are ascribed to these prior information equations.

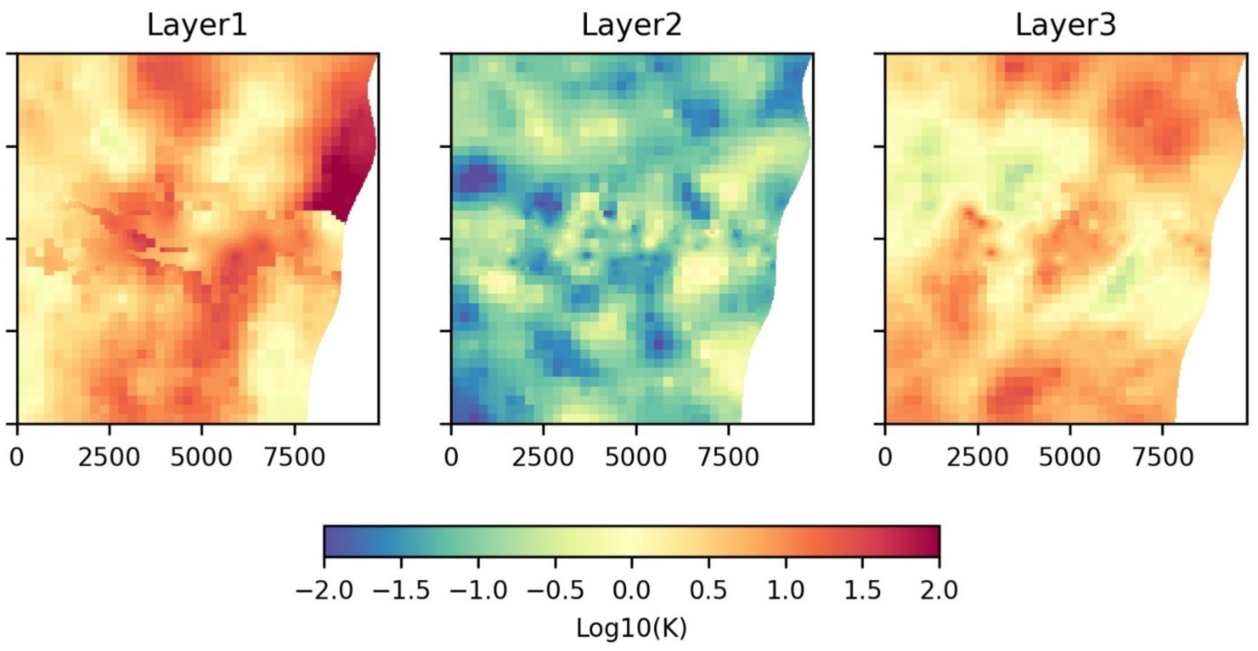


Figure 8 - Spatial distribution of K (coloured using log10 scale) in each layer from the parameter set used to generate "reality".

gmdsi

Groundwater Modelling Decision Support Initiative

gmdsi.org