



OLPROC

Processing Observations Made Easy

A GMDSI tutorial

by Rui Hugman and John Doherty



PUBLISHED BY

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

DISCLAIMER

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

PREFACE

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

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

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

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

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

Rui Hugman

John Doherty

CONTENTS

1. Introduction.....	5
1.1 What is OLPROC and Why Should I care?	5
Time-Interpolation of Model Outputs to Field Measurement Times	5
Temporal-Differences as Observations	6
Secondary Model Outcomes	7
PEST Input Dataset Construction	8
2. Background.....	10
2.1 The Groundwater Model	10
2.2 The Field Measurements	10
2.3 The History-Matching Strategy.....	11
2.4 Site Sample Files.....	11
2.5 MODFLOW 6 OBS Output Files.....	12
3. OLPROC Tutorial.....	13
3.1 Building a Simple OLPROC Control File	13
The GENERAL Block.....	13
The MEAS_SSF_FILES Block	14
The MF6_OBS_FILES Block.....	15
The OUTPUT Block	16
3.2 Running OLPROC	16
Running OLPROC in Postprocessor Mode	16
Running OLPROC in Constructor Mode.....	17
Constructor Mode: The History-Matching Dataset.....	18
Constructor Mode: PEST Instruction Files.....	18
Constructor Mode: The Partial PEST Control File	19
3.3 Observation Weights.....	19
3.4 Secondary Model Outcomes.....	21
3.5 Constructing a Full PEST Control File.....	24
Merge the Control Files.....	25
The Model-Run Batch File	26

1. INTRODUCTION

"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 is a model dancing partner. Its 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".

The current document and accompanying files offer a gentle introduction to OLPROC. First, they provide a brief overview of what OLPROC can do, and why that is useful in a parameter estimation context. Then, they demonstrate how to set up OLPROC control files to construct part of a PEST control file, starting from a dataset of measurement data files and corresponding MODFLOW 6 output files. The same OLPROC control files are later used with OLPROC as a model-output postprocessor; this is its role when model runs are undertaken by PEST.

This demonstration does not comprise a comprehensive review of all functionality available in OLPROC. Nevertheless, it should illuminate the basics of OLPROC usage. At the same time, it provides some insights into how to configure OLPROC for use in your own modelling context.

As well as OLPROC, this tutorial makes use of several other programs, such as PLPROC, PESTCHEK, PEST and MODFLOW 6. To make this tutorial easy, executable versions of programs that are needed for its completion are provided in the tutorial folder itself. However, executable versions of OLPROC and other PEST utilities can be [downloaded from the PEST web site](#). An executable version of MODFLOW 6 can be obtained [from the USGS website](#). Before commencing this tutorial, make sure that the executable version of the programs (the *.exe files) are copied to your machine. Ideally, they should be stored in a folder that is cited in your computer's PATH environment variable. Alternatively, it can simply be copied to your working folder.

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

1.1 What is OLPROC and Why Should I care?

There are several challenges where PEST is used to history-match a model against field measurements. OLPROC provides solutions to some of these challenges.

The following section provides a general overview of what OLPROC can do and how it is relevant in a PEST-controlled process. Applied examples of how to accomplish these tasks with OLPROC are demonstrated in Chapter 3.

Time-Interpolation of Model Outputs to Field Measurement Times

Put simply, when PEST supervises history-matching, it adjusts model parameters in order to reduce differences between field measurements (e.g., of groundwater levels) and their model-generated counterparts.

If a model is transient, then model outputs should be compared to field measurements at times that correspond to those at which the latter were made. In some cases, this can be accomplished by simply ensuring that the model records outputs at every field measurement time. However, if there are short intervals between field measurements over a very long period, this can lead to impractically long model run times. On the other hand (depending on the simulator) it may unnecessarily complicate model setup. Other strategies may consider using only a portion of available field measurements during history-matching, or perhaps using the average value of field measurements over a given time-interval (e.g., the average groundwater level every month). Unfortunately, both of these strategies may sacrifice information available in the field dataset.

An alternative approach is to time-interpolate model outputs to field measurement times. OLPROC accomplishes this when run as a model postprocessor. Figure 1 illustrates this. In this example, OLPROC reads model outputs (blue diamonds) and then time-interpolates these outputs to approximate values at times which correspond to those at which field measurements were made (red crosses and orange dots, respectively).

To accomplish this, all that OLPROC needs are model output files and field measurements in specific formats, along with a set of instructions on how to process these files. Examples of these instructions are provided later in this tutorial.

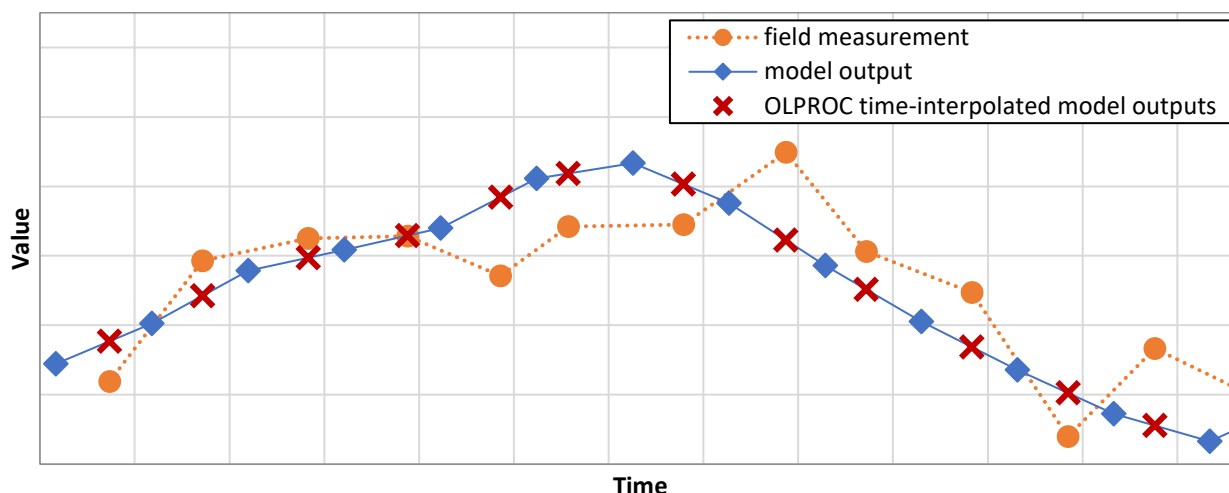


Figure 1 – Comparison of field data, model outputs and OLPROC time-interpolated model outputs. Note that time-interpolated model outputs (red crosses) coincide with field measurement times (orange circles).

Temporal-Differences as Observations

In this tutorial (and in OLPROC documentation), we denote a “temporal-difference” observation as the difference between an actual observation and a reference value of that same observation. When history-matching a transient model, these temporal-difference observations are often richer in information pertaining to certain parameters (particularly storage parameters) than raw observations (also referred to as “absolute” observations herein). Because of this, it is often useful to include them in a history-matching dataset, along with the absolute observations themselves

When deployed as a model postprocessor, OLPROC can automate temporal differencing of both measured and model-generated time series. In all cases, OLPROC employs the first value recorded in a time series as the reference value. Temporal differences are therefore calculated for each time series as the difference between the value at a given time and the first value recorded in the series.

Figure 2 shows an example of a time series of measured or modelled values together with the corresponding time-differenced time series calculated by OLPROC. Values from both, or either, of a

measurement time series could be used as history-matching targets; they are matched to their model-generated native of time-differenced series. As stated above, the use of both of these series maximizes the ability of the history matching process to extract information from field data so that it can inform model parameters. However, appropriate weighting of the absolute and difference time series should be considered; when providing weights, recall that the absolute values may be orders of magnitude larger than temporal-differenced. Each of these time series should be weighted for approximately equal visibility in the initial objective function.

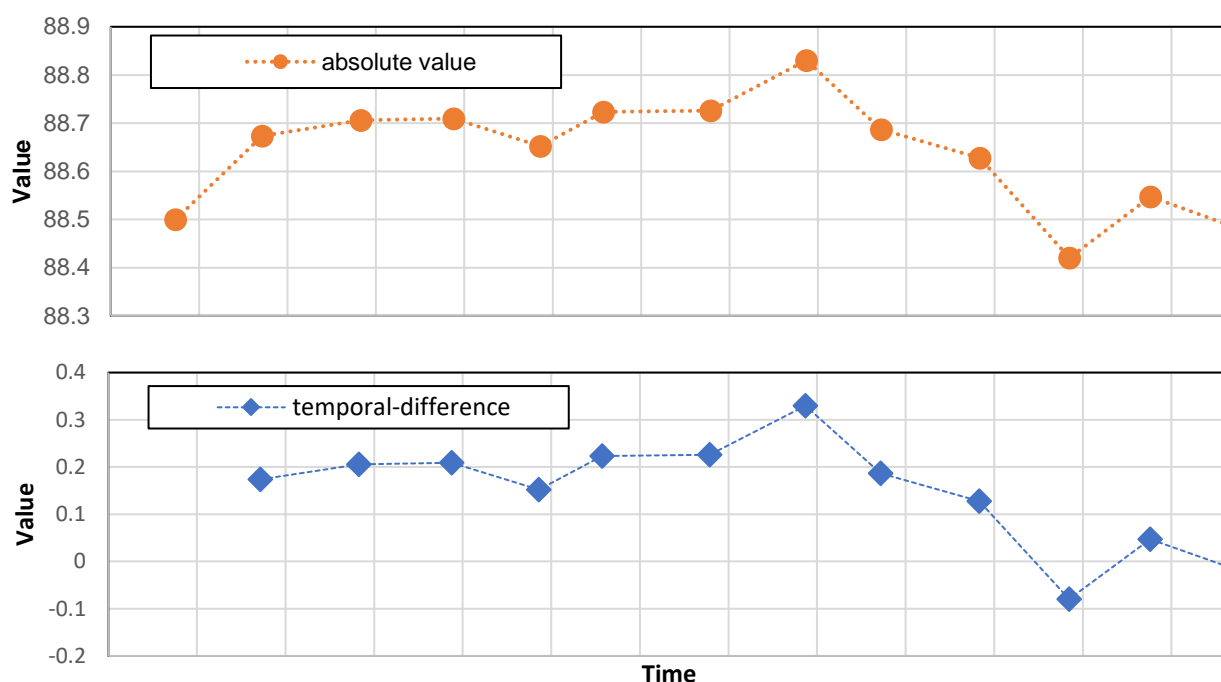


Figure 2 – Example of time series of absolute values of an observation (top) and temporal-differences of the same time series as calculated by OLPROC (bottom).

Again, all that OLPROC requires to accomplish this is a set of model output and field measurement files in appropriate formats, along with a set of instructions on how to process them. These will be described further along in the tutorial.

Secondary Model Outcomes

There may be occasions where a modeller wishes to match field measurements with quantities that are calculated from model outputs, rather than the direct outputs themselves. For example, spatial interpolation may need to be performed from a grid cell centre to the location of a measurement well; a change of scale may be required; and/or a field measurements may complement a function of several model output time series rather than a single time series.

As a model postprocessor, OLPROC provides the ability to calculate “secondary” model outcomes from direct (or “primary”) model outputs. These can then be matched to appropriate field measurements. Consider, for example head differences between vertically separated points; these differences (and their temporal variation) are often rich in information on the vertical hydraulic conductivity of an aquitard. In the same way as for temporal differences described above, , it is the nature of numerical models that they are often better at calculating differences between system states and fluxes rather than direct values of system states and fluxes. This is because model-calculated differences often suffer less from biases induced by model simplifications and defects than do absolutes. Therefore, the matching of temporal and spatial differences between model outputs to their observed counterparts can often reduce bias in the values of estimated parameters (and in the

predictions to which they are sensitive). However, few groundwater modelling codes provide in-built capabilities to set up these types of customized observations. OLPROC provides an alternative.

Figure 3 shows an example of two primary model output time series and the corresponding secondary model time series as calculated by OLPROC. The secondary model time series is calculated as the difference between values of the two primary time series (heads in this case), for every time at which the model records them. This secondary model outcome can undergo temporal interpolation by OLPROC so that it can be compared with a user-supplied time series of spatial model differences. When run as a model postprocessor, OLPROC calculates this secondary model outcome after a model run. (Note that OLPROC cannot calculate temporal differences between field-measured time series, because, unlike model-generated time series, it makes no assumption that field-measured time series employ the same time base.)

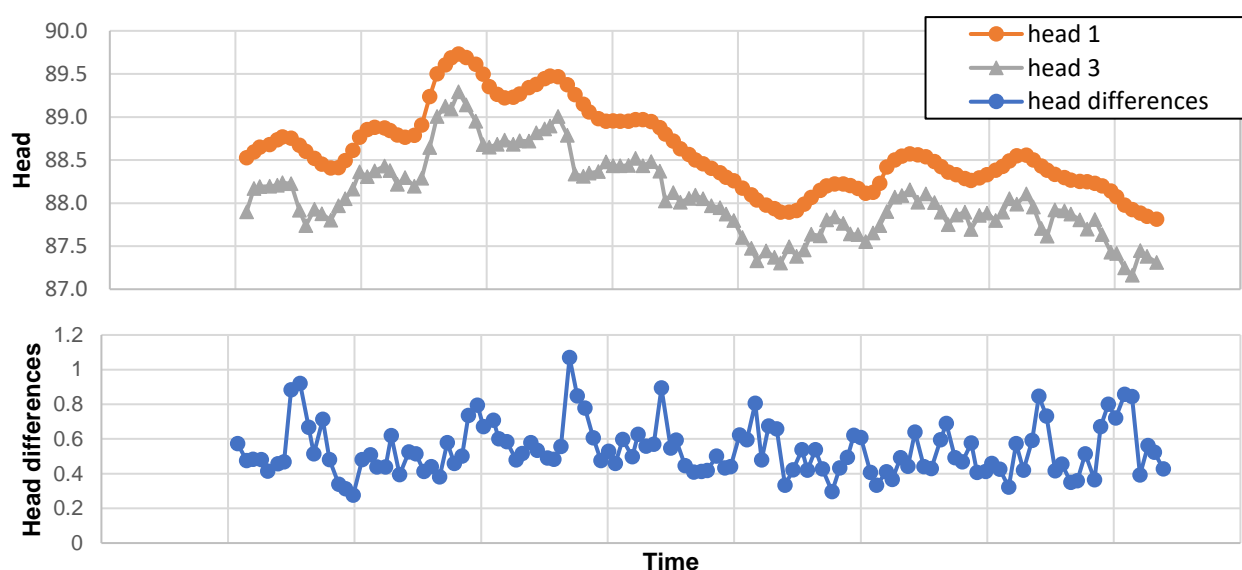


Figure 3 – Example of two head observation time series (top) and the respective head differences as calculated by OLPROC (bottom).

OLPROC treats a secondary model output in the same way that it treats a primary model output. Just as it does for primary model outputs, OLPROC time-interpolates secondary model outcomes to field measurement times, as well as calculating time-differences of secondary model outputs with respect to a reference time. Once again, all OLPROC requires are a set of model output and field measurement files in appropriate formats, along with a set of instructions on how to process them, with this processing including the calculation of secondary model outcomes if desired. These will be described further along in the tutorial.

PEST Input Dataset Construction

Preparation of an input dataset for PEST so that it can supervise the calibration of a model can be an arduous task. This is particularly so when dealing with large observation datasets. OLPROC drastically reduces the burden of constructing a PEST input dataset in which OLPROC itself is included as a model postprocessor.

A PEST input dataset is comprised of a PEST control file, one or a number of instruction files and one or a number of template files. OLPROC assists with the construction of instruction files and those parts of a PEST control file which pertain to observation data. When preparing a PEST control file, all observations need to be listed, along with complimentary information such as each observation's unique name, value, weight, and group. This needs to be done for each absolute and time-differenced value for each measurement time series that comprises the history-matching dataset. Then, the

corresponding instruction files (these are the files that “instruct” PEST on how to read model output files and extract observation values after each model run) need to be prepared. An instruction file needs to be prepared for each relevant model output file. As you can imagine, the effort to accomplish all this can quickly add up. Thankfully OLPROC can do most of it for us.

As has been mentioned previously, the PEST dataset that OLPROC constructs assumes that PEST will be running a model that includes OLPROC as a postprocessor. Therefore, to use OLPROC as a PEST input dataset constructor, OLPROC must first be set up as a model postprocessor. As has also been mentioned, this requires that OLPROC be provided with the model output files and corresponding field measurement datasets in specific formats (which will be described further along in the tutorial). Thus, when preparing a PEST input dataset, the model has to have been run beforehand in the same manner in which it will be run during the PEST controlled process.

When run in construction mode, OLPROC does several things:

- It creates a measurement dataset from field measurements that coincide with the model simulation period and/or a user-defined history-matching period.
- It creates the model-generated counterpart to the “calibration dataset to match the field measurement dataset. In this “model-calculated calibration dataset”, model outputs are time-interpolated to the times of field measurements; the model-calculated dataset can include absolute, time-difference and secondary model outcomes.
- OLPROC also writes a partial PEST control file containing the *observation groups*, *observation data* and *model input/output* sections of a complete PEST control file.
- The values of all field measurements and the user-assigned weights are referenced in the *observation groups* and *observation data* sections of the partial PEST control file.
- OLPROC also writes instruction files that read files. written by OLPROC when it is run as a model postprocessor.
- All of the instruction files and corresponding OLPROC output files are referenced in the *model input/output* section of the partial PEST control file.

The partial PEST control file that OLPROC constructs is neither complete, nor devoid of the need for further manual editing. You will need to update the remaining sections of the PEST control file or blend the contents of an OLPROC-produced partial PEST control file with those of an existing PEST control file. However, that is about as difficult as the process gets.

2. BACKGROUND

This tutorial demonstrates a simple OLPROC setup. This would commonly comprise part of a PEST workflow in which OLPROC is used to (1) build a PEST control file and then (2) post-process model outputs for PEST during model calibration. Although the workflow demonstrated herein is specific to MODFLOW 6, the concepts can be easily extended to other models and/or file structures.

Although not strictly required to complete the tutorial, it is assumed that you are familiar with PEST and MODFLOW 6 file structures.

2.1 The Groundwater Model

The model is entirely synthetic (Figure 4). The synthetic system that it simulates is multi-layered. It is comprised of a shallow unconfined aquifer, an aquitard, and an underlying confined aquifer. Several streams traverse the area, flowing from west to east. This is not really relevant to this tutorial. It simply adds some colour.

Relevant model files are provided in the tutorial folder. These include MODFLOW 6 generated output files (*heads_obs.csv*), as well as files containing field measurement data (*obs-heads1.ssf*, *obs-heads3.ssf* and *obs-deadiff.ssf*). The latter are stored in Site Sample Files (SSF files). This file structure is discussed in Chapter 2.4.

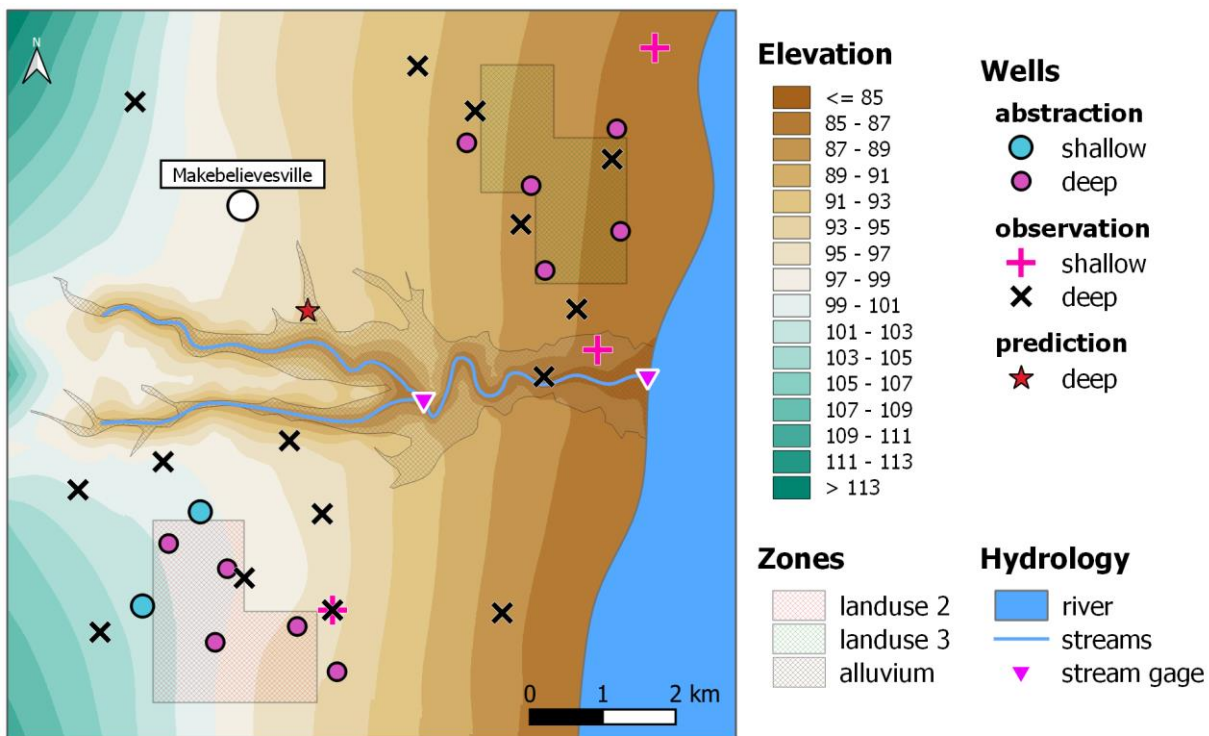


Figure 4 – Layout of the synthetic case and main features of interest.

2.2 The Field Measurements

At our synthetic site, historical measurements of hydraulic heads are only made at some wells and not at all times (Figure 5). Most monitoring wells are screened in the deeper aquifer. One nested piezometer is screened in both aquifers. Time series of measurements the nested piezometer from the upper and lower aquifer are named *H1-4429* and *H3-4429*, respectively.

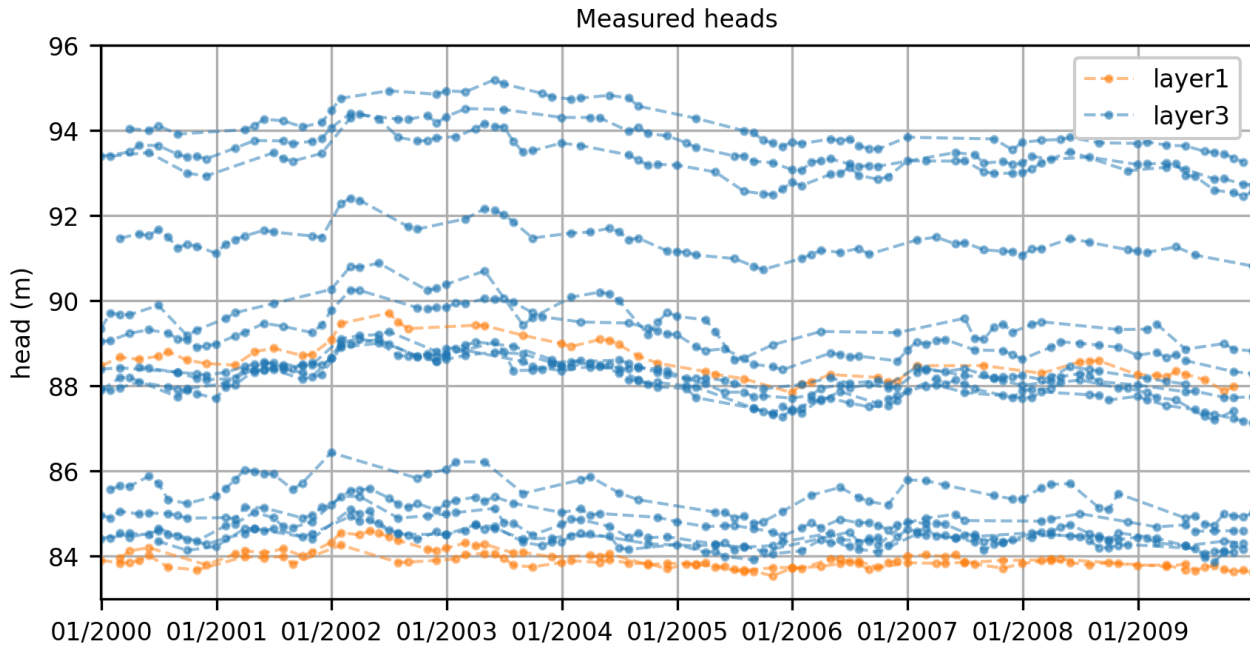


Figure 5 – Time series of measured hydraulic heads.

2.3 The History-Matching Strategy

History matching, and concomitant parameter adjustment, is to be implemented over a ten-year period (Jan 2000 to Dec 2009). The transient simulation which spans this period is preceded by a single steady state stress period. Model outputs are recorded at monthly intervals. Model output times do not necessarily match times at which field measurements were made.

PEST is tasked with minimizing a measurement objective function comprised of three components:

- Absolute values of model-calculated heads at observation wells, matched with their corresponding field-measured heads.
- Absolute differences in head between the shallow and deep aquifers calculated as secondary model outputs and compared to their measured counterparts at the single nested piezometer.
- Temporal differences for each of the time series described above. These are the difference between “the value measured at any particular time” and “the first value measured at that site”.

As you can imagine, manually building a PEST input dataset (e.g., control file and corresponding instruction files) for all these observations would require quite some effort. Fortunately, OLPROC can do most of the heavy lifting for us. All that OLPROC needs are some adequately formatted measurement data files, and some representative model output files.

2.4 Site Sample Files

Site sample files (SSF files) are also referred to in “bore sample files”. They record data gathered at various sample times at a number of locations. An SSF file contains four columns. The first column contains the names of measurement sites (i.e., targets). The second and third columns contain dates and times, respectively. The fourth column contains measurements. Dates must be formatted as dd/mm/yyyy or dd/mm/yyyy. The same format must be used consistently throughout the dataset.

The SSF file structure respects the principles of [tidy data](#). SSF files can be read by OLPROC. They are also read and/or written by many utilities in the PEST Groundwater Utilities Suite. They are described in documentation of the PEST Groundwater Utilities Suite (Part A, under “Bore Sample File”) as well as in the OLPROC manual.

OLPROC can read measurement data from SSF and/or CSV files. The current tutorial uses only SSF files. Files have been prepared; they are provided in the tutorial folder. There should be three files. These are named *obs-heads1.ssf*, *obs-heads3.ssf* and *obs-headdiff.ssf*.

Measurements of hydraulic head in the unconfined and confined aquifers are housed in *obs-heads1.ssf* and *obs-heads3.ssf*, respectively. Lastly, *obs-headdiff.ssf* contains time series of differences between head between the unconfined and the confined aquifer observed in the nested piezometer. Dates are formatted as dd/mm/yyyy.

You may be wondering why we use three SSF files, instead of just housing all the data in a single file (which would be tidier). As will become apparent later in the tutorial, OLPROC assigns data in different SSF files to different observation groups in the PEST control file which it writes. PEST can therefore record the contribution of each of these data types to the overall objective function as it lowers it. Also, a user can adjust his/her weighting scheme to ensure that data belonging to each group is visible in the overall objective function so that its information content is not ignored as parameters are estimated. (Greater flexibility in definition of observation groups is offered by OLPROC if it obtains its data from a CSV file).

2.5 MODFLOW 6 OBS Output Files

All packages of the MODFLOW 6 simulator support observation (OBS) functionality. In the input file for a particular package, a user denotes sites and data types which he/she would like recorded in an output file. Depending on the data type, the site can incorporate model-calculated quantities pertaining to one or many cells. Hence, for example, time-varying heads at a single cell centre, or time-varying fluxes through a group of cell-based boundary conditions, can be demarcated for recording. These model-calculated quantities are recorded at the end of each model time step in either a CSV or binary file. Quantities calculated for many sites can be recorded in a single file. Many such files can be recorded.

In this tutorial, OLPROC is being used to create a PEST input dataset for calibration of a MODFLOW 6 model. To create the PEST input dataset, OLPROC must have the model output files at its disposal. For the purposes of this tutorial a MODFLOW 6 model has been run, and the resulting OBS output files are provided in the tutorial folder. In the present case we are only interested in the OBS output file named *head_obs.csv*. This contains time series of heads at all of the cell centres which correspond to our field measurement sites.

Note that in the model-generated *head_obs.csv* file column headers contain observation names. More importantly, these observation names match the names of sites in the field measurement SSF files. It is important to keep this in mind when configuring a model.

3. OLPROC TUTORIAL

3.1 Building a Simple OLPROC Control File

An OLPROC user invokes various aspects of OLPROC functionality by providing it with a text input file containing a set of statements or commands which tell it what to do. This is referred to as an OLPROC control file and can easily be prepared using any text editor.

The OLPROC control file is divided into blocks. Each block begins with:

```
BEGIN blockname
```

or

```
START blockname
```

where *blockname* is the name of the block. The end of a block is identified by the text:

```
END blockname
```

Most OLPROC blocks are optional (see the OLPROC manual for descriptions of all the block types). An OLPROC control file must always contain a GENERAL. If OLPROC is to be run in constructor mode, the control file must contain an OUTPUT block. It must also contain either a MEAS_CSV_FILES block or a MEAS_SSF_FILES block (or both). It must also contain either a MODEL_SSF_FILES block or a MF6_OBS_FILES block (or both). The GENERAL block must be supplied first. Other blocks (if present) can be supplied in any order.

Comment lines and blank lines can be placed anywhere within an OLPROC control file. A comment line begins with the “#” character. Note that the “#” character must appear at the beginning of a line. If it appears elsewhere on the line, then it is interpreted as a normal character. Also note that, to avoid confusion in reading any of its input files, OLPROC does not allow a site name to begin with the “#” character.

We will start simple and slowly add complexity. Let us get started.

The GENERAL Block

- 1 Open a new blank text file in your text editor of choice. Name it *olproc1.in* and save it in your tutorial working folder. We will now proceed to fill in this file to create an OLPROC control file.
- 2 As described above, the OLPROC control file must begin with a GENERAL block. The GENERAL block provides OLPROC with information on the format used for dates, the time period spanned by the model simulation, and time period over which history-matching takes place. Type in the following block of text to delimit the GENERAL block.

```
START GENERAL  
  
END GENERAL
```

- 3 Now we can fill in the details of the GENERAL block between the START and END identifiers. First, we need to inform OLPROC of the date format used in all the input and output files. As discussed previously, all dates expressed in all measurement SSF files must follow this same protocol. Dates in the provided field measurement SSF files are formatted as *dd/mm/yyyy*. Update the GENERAL block as below:

```

START GENERAL
  date_format      = dd/mm/yyyy
END GENERAL

```

As has already been described, site sample files associate a date and a time with each measurement. However, MODFLOW 6 OBS output files report simulation time only. OLPROC internally converts these simulation times to dates and time of day. In order to do this, it must know the simulation starting time and the time units employed by the model.

- 4 Our particular MODFLOW 6 model begins with a single-day steady state time-step. This is followed by a transient simulation of the period between 00:00:00 on the 1st of January 2000 and 24:00:00 on the 31st of December 2009. The MODFLOW 6 simulation is set up to record time units as “days”. To inform OLPROC of this, we need to update the GENERAL block as below:

```

START GENERAL
  date_format      = dd/mm/yyyy

  # the 1st timestep is a 1-day steady state stress period
  # the subsequent transient stress period starts on 01/01/2000
  model_start_date  = 31/12/1999
  model_start_time  = 00:00:00

  # MODFLOW6 time-step units are setup as day.
  model_time_units  = days
END GENERAL

```

Lastly, we need to inform OLPROC that we do not want to match model generated outputs from the initial steady state time step with measured data on this occasion. Therefore, we shall set the “history matching window” to only cover the transient simulation period using the *history_match_start_date* keyword. OLPROC will disregard any field measurements or model outputs that occur outside this period.

- 5 The default values for *history_match_start_time* (00:00:00), as well as for *history_match_start_end* and *history_match_end_time* (the end of the model simulation) are acceptable for our case, so we can omit these keywords. Update the GENERAL block as below.

```

START GENERAL
  date_format      = dd/mm/yyyy

  # the 1st timestep is a 1-day steady state stress period
  # the subsequent transient stress period starts on 01/01/2000
  model_start_date  = 31/12/1999
  model_start_time  = 00:00:00

  # MODFLOW6 time-step units are setup as day.
  model_time_units  = days

  # Set the History Matching Window
  history_match_start_date = 01/01/2000
END GENERAL

```

The MEAS_SSF_FILES Block

The MEAS_SSF_FILES block provides a user with the means to nominate a set of files which OLPROC can read in order to obtain field measurements for use in the history-matching process. Along with the file name, OLPROC needs to be informed whether to use the absolute or time-difference of values contained in the file, as well as what observation group they pertain to.

To simplify its control file protocol, OLPROC decrees that all absolute measurements contained in a single SSF file be assigned to the same observation group. Likewise, all measurement time-differences are assigned to their own observation group. The name of the former group is provided by the user in the OLPROC control file. OLPROC creates a name for the latter group by affixing a “_d” suffix to this name.

- 6 Start construction of the MEAS_SSF_FILES block by inserting the following text below the GENERAL block:

```
START MEAS_SSF_FILES  
  
END MEAS_SSF_FILES
```

We will use the *file* keyword to instruct OLPROC to read the measurement files named *obs-heads1.ssf* and *obs-heads3.ssf*. Each file will be assigned to a different observation group (*heads1* and *heads3*, respectively). For each file both absolute and time-differences of measurements are to be included in the history-matching dataset. Therefore, as described above, OLPROC will automatically assign time-differences to the observation groups *heads1_d* and *heads3_d*.

- 7 Update the MEAS_SSF_FILES block as below.

```
START MEAS_SSF_FILES  
# Read the head measurements for layer 1 and layer 3  
file = obs-heads1.ssf   group=heads1   use_abs=yes use_diff=yes  
file = obs-heads3.ssf   group=heads3   use_abs=yes use_diff=yes  
END MEAS_SSF_FILES
```

The MF6_OBS_FILES Block

Now that OLPROC knows what measurement data to read, we need to provide instructions on the corresponding model outputs to read. As was mentioned above, OLPROC reads model-generated quantities from two types of file. These are:

- comma-delimited files produced by the MODFLOW 6 OBS package, and
- SSF files produced by other model postprocessors (probably belonging to the PEST Groundwater Utility suite).

In our case, because we are using MODFLOW 6 as our simulator, we will simply provide the name of a MODFLOW 6 generated CSV file. When using other simulators, model generated outputs will likely need to be converted to SSF format first. Programs from the Groundwater Utilities can perform this role. Some of them can also undertake spatial interpolation from model grid cell centres to measurements sites. (The MODFLOW 6 OBS package does not perform this function.)

The protocol for supplying filenames in an MF6_OBS_FILE block is simple. The name of a file must follow a “file” keyword. Optionally an “=” symbol can separate the filename from the *file* keyword. If the filename contains a space, then it must be surrounded by double quotes. The same block can be used to read many MODFLOW 6 generated CSV files. However, we will only be reading one.

- 8 Add the following block below the MEAS_SSF_FILES block to instruct OLPROC to read the MODFLOW 6 generated *head_obs.csv* file

```
START MF6_OBS_FILES  
file = head_obs.csv  
END MF6_OBS_FILES
```

The OUTPUT Block

Finally, the name of the partial PEST control file that OLPROC must write when run in construction mode must be provided following the mandatory *partial_pest_control_file* keyword.

- 9 Add the following block below the MF6_OBS_FILES block to instruct OLPROC to write part of a PEST control file named *partial1.pst*.

```
START OUTPUT
  partial_pest_control_file = partial1.pst
END OUTPUT
```

- 10 If desired, additional keywords can be provided in this block. These can be used to name the model batch file, as well as the folder in which OLPROC should store some of the files which it writes when run in construction mode. They do not affect how OLPROC configures observations for the PEST control file, nor how it runs in postprocessor mode.
- 11 Update the MF6_OBS_FILES block to instruct OLPROC to write the model batch file name as *runmodel.bat*. All this instruction does is defined the name of the batch file that will be included in the partial PEST control file.

```
START OUTPUT
  partial_pest_control_file = partial1.pst
  model_batch_file         = runmodel.bat
END OUTPUT
```

- 12 Lastly, update the MF6_OBS_FILES block to instruct OLPROC to write the measurement observation SSF files to a folder named *obsfiles* within your working folder. As previously mentioned, this is optional. It is merely a convenience to keep your working folder tidier. By including this instruction, OLPROC will store the SSF files of field measurement data that it generates in a separate folder. These SSF files are a subset of the original field measurement dataset and contain only the data that is included in the history-matching dataset. In our case, this means they omit any measurements that occur before 01/01/2000.
- 13 Make sure to create the new folder *obsfiles* within your working folder before running OLPROC in constructor mode.

```
START OUTPUT
  partial_pest_control_file = partial1.pst
  model_batch_file         = runmodel.bat
  obs_ssf_folder           = .\obsfiles
END OUTPUT
```

3.2 Running OLPROC

The basic information required to run OLPROC has now been prepared. We can now run OLPROC for the first time. We will do this in two parts. First, we will run OLPROC in postprocessor mode. Then we will run OLPROC in constructor mode using the same control file. It is not necessary to run OLPROC in this order; we are simply doing this to highlight the details of what OLPROC is doing in the background.

Running OLPROC in Postprocessor Mode

- 14 Open the command line in your working folder and run OLPROC in construction mode by typing `olproc olproc1.in 1` and then press <enter>. This command line tells OLPROC to read the

file *olproc1.in*, and the “1” tells OLPROC to run in postprocessor mode. (Alternatively, you can use the batch file named *run_olproc1.bat*, which is provided in the tutorial folder.)

15 If all goes well, you should see the following:

```
- file olproc1.in read ok.
- file head_obs.csv read ok.
- file obs-heads1.ssf read ok.
- file obs-heads3.ssf read ok.
- file m_obs-heads1_ssf.ssf written ok.
- file m_obs-heads3_ssf.ssf written ok.
- file m_obs-heads1_ssf_d.ssf written ok.
- file m_obs-heads3_ssf_d.ssf written ok.
```

16 The first four lines of this output show that OLPROC has read the files it was instructed to read. The last four lines show that OLPROC has written four new files. If you check your working folder, you should see them. These are the model-generated SSF files.

17 Note that all the file names begin with the prefix “m_”, followed by the name of one of the field measurement SSF files (e.g., *obs-heads1.ssf* and *obs-heads3.ssf*). Files with the “m_” prefix contain the time-interpolated model outputs to match field measurements in the history-matching time period.

18 Use a text editor to inspect the original field measurement file *obs-heads1.ssf* and the corresponding model-generated SSF file *m_obs-heads1_ssf.ssf*.

19 Note that the first line in *obs-heads1.ssf* contains data that predates the history-matching start date (01/01/2000), whilst *m_obs-heads1_ssf.ssf* does not.

20 Note that all the date & times in the model-generated SSF correspond to a date & time in the field measurement file, regardless of whether the model recorded an output at that time or not.

21 Two of the model-generated SSF files have the suffix “_d”. These are the OLPROC calculated time-differences of model outputs. Use a text editor to inspect *m_obs-heads1_ssf.ssf* and *m_obs-heads1_ssf_d.ssf*. Note that values in the latter correspond to the difference between values at each time and the first recorded value in *m_obs-heads1_ssf.ssf* (for each respective site).

These model-generated SSF files are the files that PEST will eventually see as “model outputs”. Now let us build a PEST input dataset which references these files and observations.

Running OLPROC in Constructor Mode

22 Open the command line in your working folder and run OLPROC in construction mode by typing `olproc olproc1.in 0` and then pressing <enter>. This command line tells OLPROC to read the file *olproc1.in*, and then the “0” (that is a zero, not the letter “o”) tells OLPROC to run in construction mode. (Alternatively, you can use the batch file named *run_olproc0.bat*, which is provided in the tutorial folder.)

23 If all goes well, you should see the following:

```
- file olproc1.in read ok.
- file head_obs.csv read ok.
- file obs-heads1.ssf read ok.
- file obs-heads3.ssf read ok.
- file o_obs-heads1_ssf.ssf written ok.
- file m_obs-heads1_ssf.ins written ok.
- file m_obs-heads1_ssf.ssf written ok.
- file o_obs-heads3_ssf.ssf written ok.
- file m_obs-heads3_ssf.ins written ok.
- file m_obs-heads3_ssf.ssf written ok.
- file o_obs-heads1_ssf_d.ssf written ok.
- file m_obs-heads1_ssf_d.ins written ok.
```

```

- file m_obs-heads1_ssf_d.ssf written ok.
- file o_obs-heads3_ssf_d.ssf written ok.
- file m_obs-heads3_ssf_d.ins written ok.
- file m_obs-heads3_ssf_d.ssf written ok.
- file partial1.pst written ok.

```

```

Number of observations [NOBS]      = 2140
Number of observation groups [NOBSGP] = 4
Number of instruction files [NINSFLE] = 4

```

Note that the output that OLPROC wrote when it was run in postprocessor mode is repeated, along with some additional lines. As well as re-writing the model-generated SSF files, OLPROC has done a few more things which will now be described.

Constructor Mode: The History-Matching Dataset

When run in construction mode, OLPROC writes a series of new measurement SSF files, referred to as “observation-generated SSF files”. These are the files with the prefix “o_” followed by the name of one of the user-provided measurement SSF files. The observation-generated SSF files have been recorded in the folder we specified in step 12 (*obsfiles*). Open the *obsfiles* folder to inspect the new files.

The observation-generated SSF files record values from the user-provided field measurement files that were made within the history-matching window. They omit measurements from the initial dataset that are not specifically designated for use in the history-matching process.

24 Open *obs-heads1.ssf* and *o_obs-heads1_ssf.ssf* in a text editor and compare them. Just as we saw with the model-generated SSF files (see step 18), note that the first line in *obs-heads1.ssf* contains data that predates the history-matching start date (01/01/2000), whilst *o_obs-heads1_ssf.ssf* does not.

25 If you compare *o_obs-heads1_ssf.ssf* with its model-generated counterpart (*m_obs-heads1_ssf.ssf*), you will note that all of the date & time entries match.

Just as it did for the model-generated SSF files, OLPROC calculated time-differences between field measurements within the history-matching window (see step 21). These are the observation-generated SSF files with the suffix “_d” in the file name.

26 Use a text editor to inspect *o_obs-heads1_ssf.ssf* and *o_obs-heads1_ssf_d.ssf*. Note that values in the latter correspond to the difference between values at each time and the first recorded value in *o_obs-heads1_ssf.ssf* (for each respective site).

Collectively, these OLPROC-generated SSF files contain observations which comprise the history-matching dataset. Note that observation-generated SSF files are not used directly by PEST at any point. They are merely a convenience for the user.

Constructor Mode: PEST Instruction Files

For each model-generated SSF file, OLPROC has written a corresponding PEST instruction file (files with the extension *.ins) as part of a PEST input dataset. These files instruct PEST on how to read observations from the model-generated SSF files. As will be described further on, these instruction files are referenced in the partial PEST control file which OLPROC has also written.

27 Using a text editor, inspect the instruction file named *m_obs-heads1_ssf.ins* and its model-generated SFF file counterpart *m_obs-heads1_ssf.ssf*.

28 Note that the first line in the instruction file contains the text “*pi*” followed by the delimiter “\$”. Each subsequent line contains directions which PEST must follow in order to find the next observation in the file. The PEST manual contains detailed explanations on how to prepare instruction files.

Understanding instruction files is not a requirement to using OLPROC (OLPROC handles it for you), however it is probably a good idea if you are using PEST.

Constructor Mode: The Partial PEST Control File

Last, but far from least, OLPROC has written a partial PEST control file named *partial1.pst* (see step 13). A PEST control file houses all of the variables which control how PEST will run for a given case. It also houses all the information on parameters which PEST can manipulate, as well as on the observations which PEST is comparing to model generated counterparts.

A PEST control file is divided into sections, each of which has a section header whose name begins with “*”. The partial PEST control file which OLPROC has written only contains information for the sections that pertain to observations. The remaining sections are incomplete. (Please see the PEST manual for complete specifications of a PEST control file.)

29 Open the PEST control file named *partial1.pst* and take a look. This file contains the following sections of a PEST control file:

- *observation groups*, in which observation group names are assigned to absolute and difference measurements, as defined in step 0.
- *observation data*, with observations named by site name and a sequentially increasing suffix (e.g., “_1”, “_2”, etc.) and default weights assigned to each observation.
- *model command line*, in our case, *runmodel.bat*.
- *model input/output*, containing all the OLPROC-generated model *.ins and *.ssf pairs.

It is the user’s job to add the other sections of the PEST control file to these OLPROC-written sections, or to blend the contents of an OLPROC-produced partial PEST control file with those of an existing PEST control file. When doing this, careful attention must be paid to assigning correct values to the NOBSGP, NOBS and NINSFLE variables in the “control data” section of the new PEST control file. As is explained in PEST documentation, these record the number of observation groups, number of observations and number of instruction files featured in a PEST control file. To assist the user with this task, OLPROC records the values of these variables on the screen as it ceases running in construction mode.

Merging PEST control files will be demonstrated in the “Merge the Control Files” chapter. First, we will demonstrate some more OLPROC functionality.

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

3.3 Observation Weights

OLPROC can assign weights to observations; these can vary according to observation group, observation type, and the dates and times when measurements were made.

As is described in PEST documentation, the *observation data* section of a PEST control file lists measured values to which model outputs must be fit through parameter adjustment. It also lists weights that are used in formulation of the objective function that PEST employs to denote model-to-measurement misfit.

The WEIGHTS block of an OLPROC control file contains a series of directives through which a user informs OLPROC how to calculate and assign weights to observations. This block is optional. If it is not provided, then a default value of 1.0 is ascribed to all observation weights that pertain to measurement absolutes, while a default value of 100.0 is ascribed to all observation weights that pertain to measurement differences (as you saw in the previous chapter).

There is no limit on the number of directives that can appear in the WEIGHTS section of an OLPROC control file. Each directive supersedes those which precede it. For a particular observation, weights assigned during the last directive prevail over those assigned through previous directives.

Let us begin.

30 Make a copy of the OLPROC control file *olproc1.in*. Name the copy *olproc2.in*.

31 Open *olproc2.in* in your preferred text editor.

32 Replace *partial1.pst* with *partial2.pst* in the OUTPUT block (so as not to overwrite the first partial PEST control file):

```
START OUTPUT
  partial_pest_control_file = partial2.pst
END OUTPUT
```

33 Add a WEIGHTS block by inserting the following text at the bottom of the file:

```
START WEIGHTS

END WEIGHTS
```

34 Start by assigning a weight of 0.1 to all absolute observations pertaining to the *head1* observation group, and a weight of 10 to all difference observations pertaining to the *head1* group. (There is no particular reason behind the choice of these values, they are simply a demonstration.)

35 Each line within a WEIGHTS block must begin with the string "*name=*". The name of a measurement site or observation group must follow this string. If the group name is used, then the type (*abs* or *diff*) must be specified. Update the WEIGHTS block with the following:

```
START WEIGHTS
  name = heads1 type=diff weight= 10.0
  name = heads1 type=abs weight= 0.1
END WEIGHTS
```

36 Open the command line in your working folder and run OLPROC in construction mode by typing *olproc olproc2.in 0* and then press <enter>.

37 Check the new *partial2.pst* PEST control file. Compare the observation weights with those in *partial1.pst*, and you will see that all weights associated with observations belonging to groups *heads1* and *heads1_d* have been updated.

38 Now, suppose that a greater weight is desired for measurements that occur during a specific period. (Perhaps the modeller wishes to ensure that behaviour during a drought year is replicated well by the model). This can be accomplished using the *date1*, *time1*, *date2* and *time2* keywords. (All of these are optional, see the OLPROC manual for further details and default values).

39 Assign a weight of 30.0 to all *diff* measurements in the *heads1* and *heads3* groups, between 01/10/2004 and 01/10/2006. The *date1*, *time1*, *date2* and *time2* keywords denote the start and end date and time of the period to which these higher weights will be assigned (the *time* keywords are not strictly necessary in this case, as the OLPROC default *time* value of 00:00:00 would result in the same outcome). Update the WEIGHTS block as follows:

```

START WEIGHTS
# Set weights for heads1 group
name = heads1 type=diff weight= 10.0
name = heads1 type=abs weight= 0.1

# Set weights for heads1 and heads3, during specific period
name = heads1 type=diff weight= 30.0    &
      date1=01/10/2004 time1 = 00:00:00 &
      date2=30/09/2006 time2 = 23:59:59
name = heads3 type=diff weight= 30.0    &
      date1=01/10/2004 time1 = 00:00:00 &
      date2=30/09/2006 time2 = 23:59:59
END WEIGHTS

```

40 Open a command line window in your working folder and run OLPROC in construction mode by typing `olproc olproc2.in 0` and then press <enter>.

41 You should see the following on your screen:

```

- file olproc2.in read ok.
- file head_obs.csv read ok.
- file obs-heads1.ssf read ok.
- file obs-heads3.ssf read ok.
- file o_obs-heads1_ssf.ssf written ok.
- file m_obs-heads1_ssf.ins written ok.
- file m_obs-heads1_ssf.ssf written ok.
- file o_obs-heads3_ssf.ssf written ok.
- file m_obs-heads3_ssf.ins written ok.
- file m_obs-heads3_ssf.ssf written ok.
- file o_obs-heads1_ssf_d.ssf written ok.
- file m_obs-heads1_ssf_d.ins written ok.
- file m_obs-heads1_ssf_d.ssf written ok.
- file o_obs-heads3_ssf_d.ssf written ok.
- file m_obs-heads3_ssf_d.ins written ok.
- file m_obs-heads3_ssf_d.ssf written ok.
- file partial1.pst written ok.

Number of observations [NOBS]      = 2140
Number of observation groups [NOBSGP] = 4
Number of instruction files [NINSFLE] = 4

```

42 Check *partial2.pst* to see the changes. As you scroll through the observations, you should see that some of the observations in group *heads1_d* and *heads3_d* now have weights of 30.0.

You will have noticed that we are assigning weights twice to time-difference observations of *heads1*. Therefore, we have to ensure that the last instruction (from top to bottom) provides the final weight value that is associated with observations appearing in the PEST control file. To ensure this, it is generally a good idea to start with broad weight assignments at the top of the WEIGHTS block and work downwards to targeted weights assignments at the bottom of the block.

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

3.4 Secondary Model Outcomes

OLPROC supports calculation of “secondary model outcomes”. These can be matched with field measurements during the history-matching process. Examples include differencing of time series (as will be demonstrated here), spatial interpolation from grid cell centres to the location of a measurement well and introduction of a change of scale or offset.

Secondary model outcomes are calculated from “primary” model outcomes and other secondary model outcomes. Primary model outcomes refer to quantities that are featured in the MODFLOW 6 OBS output files or in model SSF files that OLPROC reads. Each of these files records time series of observations pertaining to one or more sites. A secondary model outcome associates a time series with a site whose name is specified by the user. Terms of this new time series are calculated from terms of existing time series using an equation. This same equation is applied to all terms comprising the series from which they are calculated. The site with which the new series is associated can be new, and hence named by the user through the equation itself. Alternatively, the new series can be ascribed to an existing site; in the latter case, the existing time series associated with that site is overwritten.

We will now demonstrate how to set up a SECONDARY_MODEL_OUTCOMES block to calculate difference between values from two time series. In this case, the two time series are observations of head at different depths in a nested piezometer. If you open the original measurement SSF files *obs-heads1.ssf* and *obs-heads3.ssf*, you will find entries for observation sites *H1-4429* and *H3-4429*, respectively. (The prefix “H1-” and “H3-” refer to layer 1 and layer 3, respectively; “4429” refers to the piezometer.)

Let us begin.

43 Make a copy of the OLPROC control file *olproc2.in*. Name the copy *olproc3.in*.

44 Open *olproc3.in* in your preferred text editor.

45 Replace *partial2.pst* with *partial3.pst* in the OUTPUT block of *olproc3.in* with the following text (so as not to overwrite the first partial PEST control file):

```
START OUTPUT
  partial_pest_control_file = partial3.pst
END OUTPUT
```

46 Add a SECONDARY_MODEL_OUTCOMES block by inserting the following text at the bottom of the file:

```
START SECONDARY_MODEL_OUTCOMES

END SECONDARY_MODEL_OUTCOMES
```

Each line in a SECONDARY_MODEL_OUTCOMES block must contain an assignment equation. The equation can be of arbitrary complexity. Non-numeric variables featured on the right of an equation must be the names of sites that have been previously defined in MODFLOW 6 OBS output files, model SSF files, or in previous equations provided in the SECONDARY_MODEL_OUTCOMES block. As stated above, the equation is applied to all terms in the time series associated with these sites to generate corresponding terms in the new time series.

The term on the left of an equation can pertain to an existing site. Alternatively, it can define a new site. If it pertains to an existing site, then all terms of the time series associated with that site are overwritten. If it pertains to a new site, the OLPROC name limit of 20 characters must be respected when naming the new site. OLPROC also requires the name of a site to be surrounded by quotes if it contains a character that may be confused with a numerical operator.

As you may recall, our site names (*H1-4429* and *H3-4429*) include the character “-”, which is also the differencing numerical operator. These names must therefore be surrounded by quotes when used in an equation.

Let us continue.

- 47 Create a secondary model outcome named *dh4429* from the difference between *H1-4429* and *H3-4429* by updating the SECONDARY_MODEL_OUTCOMES block in the following way:

```
START SECONDARY_MODEL_OUTCOMES
  dh4429 = "H1-4429" - "H3-4429"
END SECONDARY_MODEL_OUTCOMES
```

- 48 That is all there is to it (well, almost!). Transfer to the command line window in your working folder and run OLPROC in construction mode by typing `olproc olproc3.in 0`; then press <enter>.

- 49 You should see the following on your screen:

```
- file olproc3.in read ok.
- file head_obs.csv read ok.
- file obs-heads1.ssf read ok.
- file obs-heads3.ssf read ok.
- model observation equations evaluated ok.
- file o_obs-heads1_ssf.ssf written ok.
- file m_obs-heads1_ssf.ins written ok.
- file m_obs-heads1_ssf.ssf written ok.
- file o_obs-heads3_ssf.ssf written ok.
- file m_obs-heads3_ssf.ins written ok.
- file m_obs-heads3_ssf.ssf written ok.
- file o_obs-heads1_ssf_d.ssf written ok.
- file m_obs-heads1_ssf_d.ins written ok.
- file m_obs-heads1_ssf_d.ssf written ok.
- file o_obs-heads3_ssf_d.ssf written ok.
- file m_obs-heads3_ssf_d.ins written ok.
- file m_obs-heads3_ssf_d.ssf written ok.
- file partial3.pst written ok.
```

```
Number of observations [NOBS]      = 2140
Number of observation groups [NOBSGP] = 4
Number of instruction files [NINSFLE] = 4
```

- 50 Compare this to when you ran *olproc2.in* (see step 41). OLPROC reports the same number of observations. But there should be more!

- 51 If you inspect the new *partial3.pst* PEST control file, you will find that it is identical to *partial2.pst*. This is because OLPROC will only record observations in the PEST control file if there is a corresponding measurement site with the same name. Such a name has not yet provided!

It is up to the user to prepare measurement time series that correspond to model primary or secondary outputs. It is these measurements that define a calibration dataset, and hence the contents of OLPROC outputs. For the present tutorial, we need a measurement time series of differences between head at *H1-4429* and heads at *H3-4429*. A file of head difference measurements named *obs-headdiff.ssf* has been prepared for you; it can be found in the tutorial folder.

- 52 Update the MEAS_SSF_FILES block so that it tells OLPROC to read *obs-headdiff.ssf*. Use both absolute and difference observations. Name the observation group *headdiff*.

```
START MEAS_SSF_FILES
# Read the head measurements for layer 1 and layer 3
file = obs-heads1.ssf  group=heads1  use_abs=yes use_diff=yes
file = obs-heads3.ssf  group=heads3  use_abs=yes use_diff=yes

# headdiff is the head difference between layer 1 and 3 at the same site
file = obs-headdiff.ssf group=headdiff use_abs=yes use_diff=yes
END MEAS_SSF_FILES
```

53 In practice, you might need to update weighting schemes as well. For the purposes of the tutorial, we will simply leave them with the default values.

54 Try running OLPROC again. Open the command line in your working folder and run OLPROC in construction mode by typing `olproc olproc3.in 0` and then press <enter>.

55 You should see the following on your screen:

```
- file olproc3.in read ok.
- file head_obs.csv read ok.
- file obs-heads1.ssf read ok.
- file obs-heads3.ssf read ok.
- file obs-headdiff.ssf read ok.
- model observation equations evaluated ok.
- file o_obs-heads1_ssf.ssf written ok.
- file m_obs-heads1_ssf.ins written ok.
- file m_obs-heads1_ssf.ssf written ok.
- file o_obs-heads3_ssf.ssf written ok.
- file m_obs-heads3_ssf.ins written ok.
- file m_obs-heads3_ssf.ssf written ok.
- file o_obs-headdiff_ssf.ssf written ok.
- file m_obs-headdiff_ssf.ins written ok.
- file m_obs-headdiff_ssf.ssf written ok.
- file o_obs-heads1_ssf_d.ssf written ok.
- file m_obs-heads1_ssf_d.ins written ok.
- file m_obs-heads1_ssf_d.ssf written ok.
- file o_obs-heads3_ssf_d.ssf written ok.
- file m_obs-heads3_ssf_d.ins written ok.
- file m_obs-heads3_ssf_d.ssf written ok.
- file o_obs-headdiff_ssf_d.ssf written ok.
- file m_obs-headdiff_ssf_d.ins written ok.
- file m_obs-headdiff_ssf_d.ssf written ok.
- file partial3.pst written ok.
```

```
Number of observations [NOBS]      = 2193
Number of observation groups [NOBSGP] = 6
Number of instruction files [NINSFLE] = 6
```

56 That looks better. We now have more observations and observation groups. Additionally, OLPROC has recorded two new sets of observation-generated and model-generated SSF files, and a PEST instruction file to read the latter.

57 If you inspect the new *partial3.pst* PEST control file you should see two new observation groups. If you scroll to the end of the file, you should see observations corresponding to the *dh4429* time series as well as the associated model output and instruction files.

All files generated so far can be found in the folder *t3*.

3.5 Constructing a Full PEST Control File

The tutorial thus far demonstrates the basics of generating a partial PEST control file, starting from a dataset of measurements and corresponding model output files. In practice, *partial3.pst* would now be expanded to include the other sections of a PEST control file.

There is a sub-folder in the tutorial folder named *mf6model*. This sub-folder contains the files required to run the MODFLOW 6 model described at the beginning of this document. It also contains a set of PEST template (*.tpl) files and another partial PEST control file named *partial_param.pst*. This partial control file contains the PEST control file sections which are missing from the partial control files

constructed by OLPROC. (The parameters in *partial_param.pst* are pilot points. Their values are assigned to the conductance of a general-head boundary condition cells in the MODFLOW 6 model using PLPROC. This is [described in other GMSI tutorials.](#))

We will now merge these two partial PEST control files which, along with the instruction files and template files, comprise a complete PEST input dataset.

Merge the Control Files

Let us begin.

58 Open *partial3.pst* in a text editor.

59 Make a copy of *partial_param.pst*. Name it *full.pst* and open it in a text editor.

60 From *partial3.pst*, copy the **observation data*, **observation group* and **model command line* sections and paste them into the corresponding sections in *full.pst*.

61 Next, from *partial3.pst*, copy the lines in the **model input/output* section. These are all the lines below the line containing the text “*INSERT TEMPLATE AND MODEL INPUT FILES HERE*”.

62 Paste these into *full.pst*, below the lines listing the model template files. It should look something like this (lines copied from *partial3.pst* are shown in **bold**):

```
* model input/output
ghbpp.tpl          ghbpp.dat
m_obs-heads1_ssf.ins    m_obs-heads1_ssf.ssf
m_obs-heads3_ssf.ins    m_obs-heads3_ssf.ssf
m_obs-headdiff_ssf.ins  m_obs-headdiff_ssf.ssf
m_obs-heads1_ssf_d.ins  m_obs-heads1_ssf_d.ssf
m_obs-heads3_ssf_d.ins  m_obs-heads3_ssf_d.ssf
m_obs-headdiff_ssf_d.ins m_obs-headdiff_ssf_d.ssf
```

63 Lastly, we need to update the NOBS, NOBSGP and NINSFLE variables in the **control data* section. These stand for “number of observations”, “number of observation groups” and “number of instruction files”, respectively.

64 Scroll up to the top of *full.pst*. The **control data* section should currently look like this:

```
* control data
restart estimation
2728      0      1      0      0
1 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
```

65 The positions of the NOBS, NOBSGP and NINSFLE variables are shown below.

```
* control data
restart estimation
2728      NOBS      1      0      NOBSGP
1 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
```

66 Recall that when OLPROC ran in construction mode it wrote the values of these variables to the screen (see step 55). You can also retrieve them by simply counting the number of lines in each of the respective control file sections. Update the values of NOBS, NOBSGP and NINSFLE in *full.pst* accordingly. It should look something like this (the three updated values are in **bold**):

```
* control data
restart estimation
2728      2193      1      0      6
1 6 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
```

And that is it. You now have a complete PEST control file.

67 Run PESTCHEK to verify its integrity. Open the command line, type *pestchek full.pst* and press <enter>. If all went well, you should see the following appear on your screen:

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

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

Lastly, we need to set up a model run batch file that includes OLPROC as a model postprocessor.

The Model-Run Batch File

In PEST managed model runs, OLPROC would be called in postprocessor mode through the model batch file. The user needs to ensure that the command to run OLPROC is included in the model batch file; the command to run OLPROC should follow the command to run the model and any model postprocessors that write SSF files that OLPROC may read.

In the current case, what PEST considers to be “the model” is composed of three parts. PLPROC is first called to interpolate parameter values from pilot points to the groundwater model cells. Then, the groundwater model is simulated. Lastly, OLPROC postprocesses the model outputs. PEST intervenes in this process at two stages: first it writes values to the pilot point files, second it reads values from OLPROC generated output SSF files.

Let us prepare a batch file which PEST will then use to run the model.

68 Copy all the files in the *mf6model* folder into your working folder.

69 Create a new text file named *runmodel.bat*.

70 Type the following into *runmodel.bat* and then save the file.

```
plproc plproc_ghb.dat
mf6 mfsim.nam
olproc olproc3.in 1
pause
```

71 The first line runs PLPROC using a previously prepared PLPROC script file (see [other GMSI tutorials](#) for details).

- 72 The second line runs the MODLFOW 6 model.
- 73 The third line runs OLPROC in postprocessor mode using the *olproc3.in* control file.
- 74 The last line keeps the command line window open when the process finishes (this line should not be included in a batch file provided to PEST; it is purely for the sake of the tutorial so that you can see the outcome of running the batch file).
- 75 Now execute *runmodel.bat* (double-click on it or type its name at the screen prompt). It should take about 10 seconds for the model to run. When complete, you should see the following:
- file olproc3.in read ok.
 - file head_obs.csv read ok.
 - file obs-heads1.ssf read ok.
 - file obs-heads3.ssf read ok.
 - file obs-headdiff.ssf read ok.
 - model observation equations evaluated ok.
 - file m_obs-heads1_ssf.ssf written ok.
 - file m_obs-heads3_ssf.ssf written ok.
 - file m_obs-headdiff_ssf.ssf written ok.
 - file m_obs-heads1_ssf_d.ssf written ok.
 - file m_obs-heads3_ssf_d.ssf written ok.
 - file m_obs-headdiff_ssf_d.ssf written ok.
- 76 Check your working folder. You should see the SSF files with a “m_” prefix have been updated. If you inspect their contents you should see that they differ from the corresponding files we created earlier. (This is because the model uses a different set of parameters from those which were used to generate the files we were working with previously.)
- 77 If you desire, you can now use *full.pst* to run PEST. The value for NOPTMAX is set to zero, therefore PEST will only run the model once and stop (this should take around 10 seconds). To do so, open the command line, type *pest full.pst* and press <enter>. PEST will call the model run batch file. Once finished it will output the summary of observation residuals to the command line window and write several output files to your working folder.

Congratulations, you have just emulated a (small) part of what PEST would do several thousand times as it calibrates a model. Hopefully, this tutorial has provided some insights into some of OLPROC's functionality, as well as how to use it for PEST run setup.

The case demonstrated here was relatively simple; this lightens the burden of learning new software. However, OLPROC can be used to set up much more complex cases than that which is demonstrated here. It can also facilitate the creative use of observations to assist calibration-based data assimilation. Other GMDSI Tutorials and Worked Examples may explore examples of more complex and creative setups.

All files generated so far can be found in the folder *t4*.



gmdsi.org

CRICOS NO 00114A

BHP



Flinders
UNIVERSITY



NATIONAL CENTRE FOR
GROUNDWATER
RESEARCH AND TRAINING

RioTinto