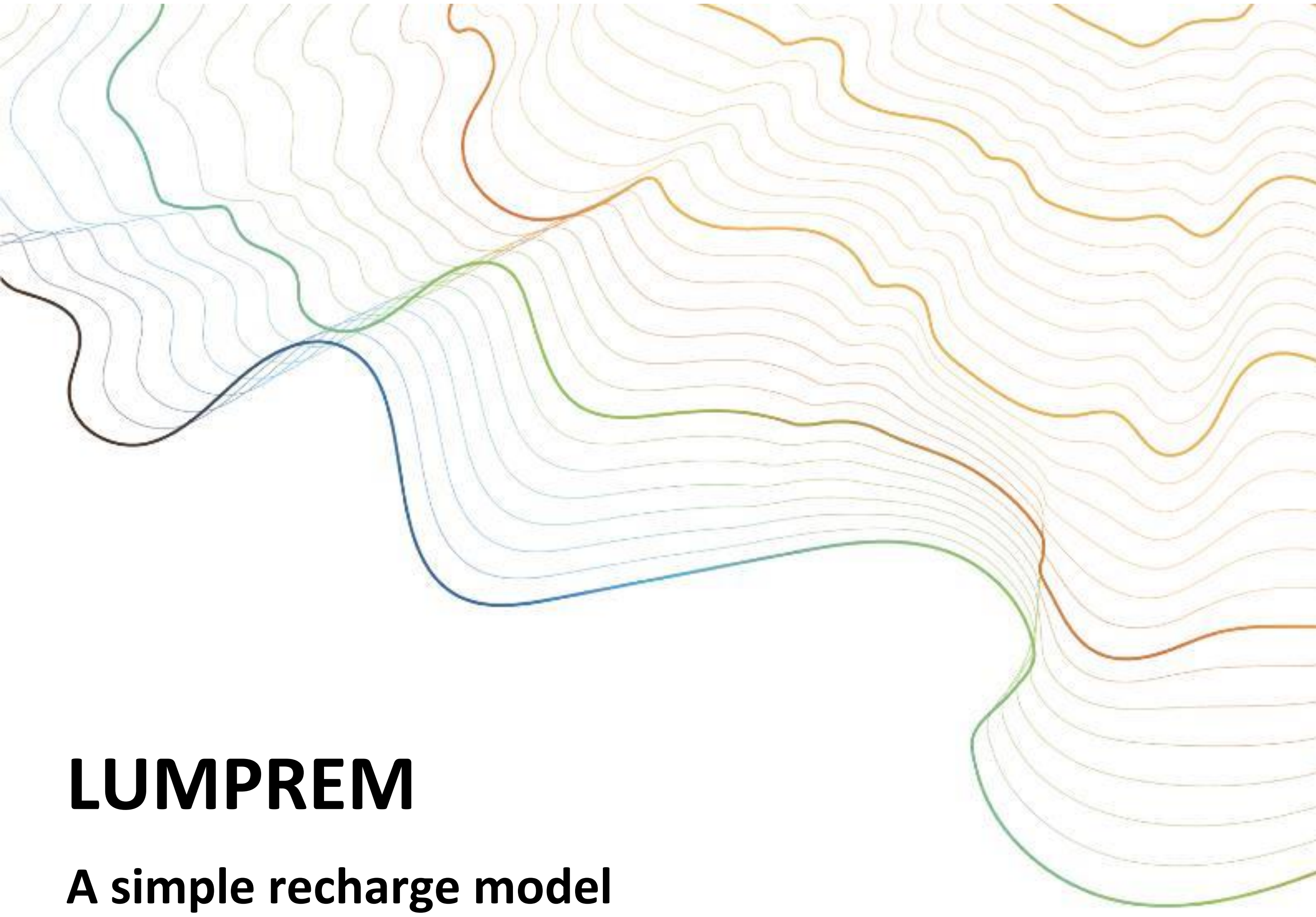

LUMPREM

A simple recharge model

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

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 contexts.

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 the same synthetic case and are based on the same simulator (MODFLOW 6). Utility software from the PEST suite is extensively used 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. That being said, 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 provides a tutorial on how to set up and use a simple lumped parameter recharge model (named LUMPREM) in conjunction with a groundwater model. The tutorial will cover how to set up LUMPREM models and how to convert LUMPREM outputs into inputs for a MODFLOW 6 model.

LUMPREM generates time series of groundwater recharge using daily rainfall and potential evaporation as input. It can also be used to simulate crop water demand and residual potential evaporation, i.e. the potential evaporation that is available to the groundwater system after the demands of the unsaturated zone have been met. As well as this, it can generate proxy time series of groundwater levels that can be applied to boundaries of a groundwater model. The concepts and processes simulated by LUMPREM are described in the software's documentation, which we recommend reading (go on, we will wait). It is assumed that the reader is at least superficially familiar with LUMPREM documentation. The current document does not describe these concepts in detail; rather, it focusses on how to set up LUMPREM models for use with a groundwater model. In the present tutorial, MODFLOW 6 is our groundwater model. However, the workflow described herein can easily be adapted to other models.

This document will cover construction of LUMPREM model input files for different land-use and irrigation characteristics. It will then discuss how to use the LR2SERIES utility to translate LUMPREM-generated time series to a format that the RECHARGE, EVT and GHB packages of MODFLOW 6 can read. Lastly, it will demonstrate how to manipulate MODFLOW 6 time series files using the TS6PROC utility. A schematic overview of the workflow is shown in the diagram below.

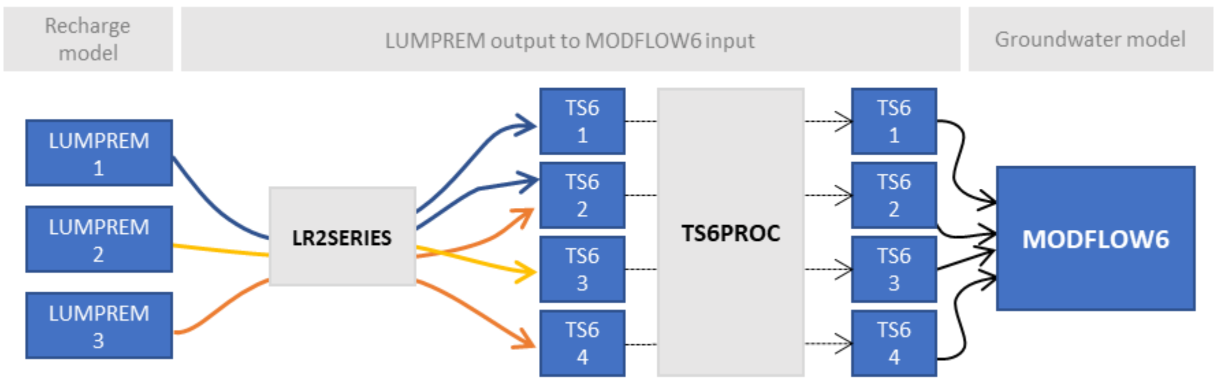

Figure 1 Diagram of a LUMPREM to MODFLOW 6 workflow.

The workflow will be demonstrated using the command line and text files. If you are partial to using Python, the Lumpyrem package can be used in setting up and interacting with LUMPREM, LR2SERIES and Flopy. Lumpyrem can be installed using pip or downloaded from GitHub. The latter is the most up to date version. Example notebooks can be found in the GitHub repository. The current tutorial does **not** cover use of Lumpyrem.

The underlying purpose of workflows described in this and other tutorials is to facilitate data assimilation to quantify and constrain the uncertainties of a groundwater model's predictions. As LUMPREM is fast and nimble, parameters that affect recharge processes can be included in history matching and uncertainty analysis without much additional computational cost. However, this requires that the workflow (from LUMPREM to groundwater model) be automated as a set of commands (i.e. batch file) that PEST/PEST++ can call during an inversion or uncertainty analysis process. During that process, the parameters that are being estimated can belong to (a) MODFLOW (b) LUMPREM and/or (c) some processing programs that manipulate time series that are passed between the two. This document will discuss how to set up the various steps of an automated workflow to run LUMPREM and prepare MODFLOW 6 input files. However, it will not discuss the use of PEST/PEST++.

1.1 The Groundwater Model

The examples discussed in the present document employ a synthetic MODFLOW 6 model that is used in other GMDSI tutorials. Our hypothetical study area includes three land-use types. Each of these land-use types has vegetation with different characteristics. Two of them represent irrigated agriculture. All irrigation water comes from groundwater abstraction. Some irrigation is seasonal, some is year-round.

The western edge of the model is represented using a general-head boundary (GHB) condition. The elevation of this GHB is assumed to vary seasonally.

Therefore, to set up our MODFLOW 6 model the following are required:

- three different time series of groundwater recharge (one for each land-use type),
- three different time series of potential evaporation (one for each land-use type),
- two different time series of groundwater abstraction (one for each irrigated land-use type),
- one time series of GHB elevation.

The MODFLOW 6 model is endowed with an initial steady-state stress-period. An average value for each of the described time series is required for this stress-period.

1.2 Before we get started

Make sure that executable versions LUMPREM, LR2SERIES and TS6PROC are copied to your machine; these are files *lumprem.exe*, *lr2series.exe* and *ts6proc.exe* For convenience, the folder in which these executables are stored should be added to your computer's PATH environment variable. Alternatively, they can be placed in the working folder for this tutorial. To make completion of this tutorial easy, these executables are provided in the *software* sub-folder.

Two input files are provided (Table 1). Make sure that these are in the tutorial folder.

Table 1 Files provided for the LUMPREM tutorial.

File name	**Description**
climate_data.csv	rainfall and potential evaporation data
lumprem_input_file_blank.in	example blank LUMPREM input file

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.

2. LUMPREM

LUMPREM can be run in one of two ways. The first is to type its name at the command line prompt. On commencement of execution, LUMPREM prompts for the name of its input control file, and for the name of the output file which it must write. Alternatively, the name of its input and output files can be supplied directly on the LUMPREM command line. An example will be demonstrated further on.

From the above it is apparent that to run LUMPREM we need to prepare an input file. We must also specify the name and location of the file in which LUMPREM outputs will be saved. The input file specifies parameters used by LUMPREM, as well as the names of other files that contain rainfall and potential evaporation time series. If vegetation characteristics and irrigation rates vary over time, these can be provided in files as well.

The relationship between LUMPREM and its input/output files is schematized below. As you can see, each LUMPREM model run requires a unique input file and produces a unique output file. Files that contain input time series (rainfall, potential evaporation, etc.) can be unique to each LUMPREM model, or shared between several LUMPREM models. In the following chapters we will demonstrate a case in which several LUMPREM models share rainfall, potential evaporation and vegetation input files, but each have different inputs for irrigation (as illustrated in the diagram below).

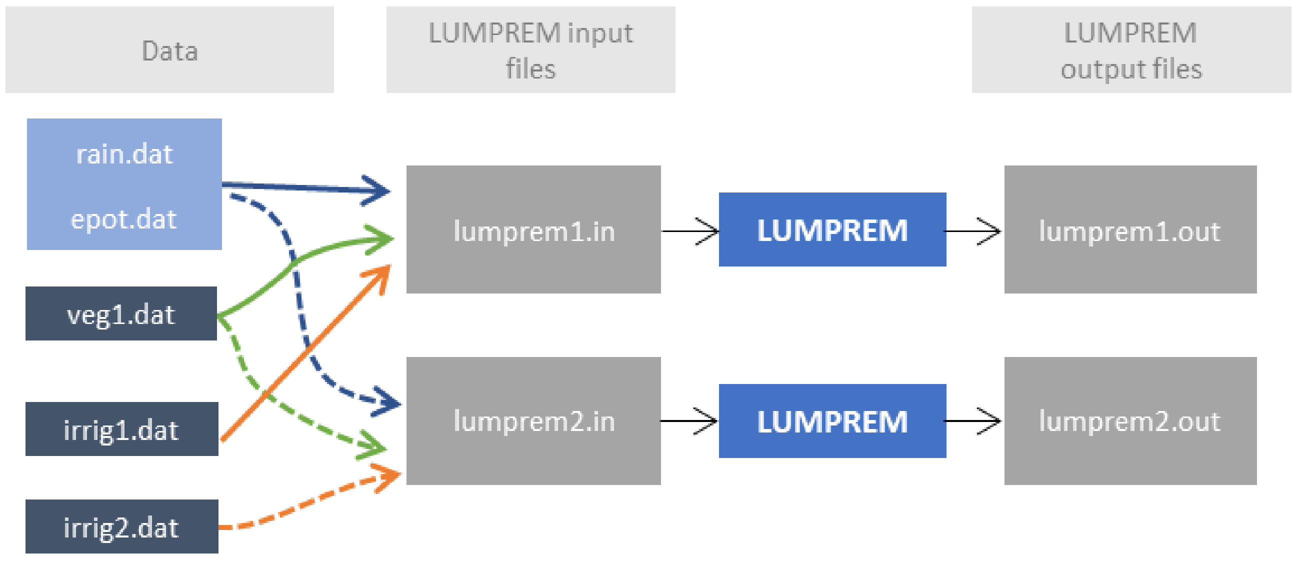

Figure 2 Diagram of a LUMPREM workflow and input/output files.

2.1 Setting up *rain* and *potential evaporation* files

Open file *climate_data.csv* using your preferred spreadsheet software (for example, Microsoft EXCEL). Take a look at the structure of this file. This is our "raw data" and is likely representative of climate data that may be available at any given site. As you can see, daily values are provided over a roughly 2-year period. Some days have missing values. Note that units are in mm/day. We will need to make use of the data in this file to create the rain and potential evaporation input files for LUMPREM.

To keep things simple, let us assume that the first day of our simulation is the 1st of January 2000 and that the last day of our simulation is 31st of December 2001. When we set up the MODFLOW 6 model, time units of metres and days will be employed. For consistency, we will therefore use metres as our LUMPREM length unit (the use of days for time is mandatory). So, our first step is to convert climate data supplied in *climate_data.csv* from units of mm/d to those of m/d. Go ahead and do this now. If you are using a spreadsheet package such as EXCEL, be careful to note that blank cells are not the same thing as zero.

1 Convert the units of rainfall and potential evaporation values in *climate_data.csv* to m/d.

Let us now create two files, one of which (*rain.dat*) will hold rainfall data, and the other of which (*epot.dat*) will hold potential evaporation data. These two files will be used by all the instances of LUMPREM that we develop in this tutorial. They can be created in any text editor or spreadsheet software. The format for *rain.dat* and *epot.dat* is quite forgiving. Each must contain two columns of data. On each line the first entry is the simulation day (an integer), while the second entry is the value corresponding to that day (a real number). Days must be provided in increasing order. Not every day needs to be represented. However, days with missing values must be removed (if they are not, LUMPREM will return an error specifying that there are insufficient entries). After removing rows with missing entries, LUMPREM will assume values on days which are not cited in the input file. Rainfall will be assumed as zero and potential evaporation as the value from the previous day.

After converting the units, copy values in the *Day* and rainfall columns of file *climate_data.csv* to a new file named *rain.dat*. The new file can be either comma or whitespace delimited. **Do not** include column name headers in this file. Do the same for data in the *Ev* column; call the new file *epot.dat*. In practice, you can give the files whatever names you like; however, these names are easy to remember.

2 Create two new files: *rain.dat* and *epot.dat*.

3 Copy the Day and rainfall (m/d) columns from *climate_data.csv* to *rain.dat.* Do not include column name headers.

4 Copy the Day and potential evaporation (m/d) columns from *climate_data.csv* to *epot.dat.* Do not include column name headers.

5 In each file, delete all rows that have missing entries (i.e. that have a value in the Day column, but no value in the second column) .

2.2 Setting up a LUMPREM input file

Let us start off slowly with just one LUMPREM model. The *lumprem_input_file_blank.in* file provided with this tutorial specifies the layout of such a file. Please read the LUMPREM manual for detailed explanations for all of its variables. Let us now work through them as we turn these specifications into a real LUMPREM input file.

6 Make a copy of *lumprem_input_file_blank.in*. Name this copy *lr_lu1.in*. This stands for "lumprem land-use 1". Again, you can actually call it whatever you like, but we recommend that you stick with this name for the sake of consistency with the tutorial.

7 Now, let us edit *lr_lu1.in*. Open the file in a text editor (for example, Notepad).

8 You will note that the file is divided into sections, denoted by a row with text preceded by an "*" (i.e. "* earth properties", "* volume to elevation", etc.). Section titles should not be edited. only the variables cited within a section can be edited. Let us start by editing the variables of the first section: **** earth properties***. We will now proceed to replace each of the variable names with a value.

9 The variable ***maxvol*** refers to the total volume of the soil moisture store. It is roughly equal to the depth of the root zone times the soil porosity. Average root depths vary with vegetation type but

are often between 0.3 to 1 m. Let us assume vegetation has a root depth of 1m, and soil porosity is on average 0.2, giving us a ***maxvol*** of 0.2. Replace "maxvol" in the file with the value 0.2.

10 ***irrigvolfrac*** represents the fraction of soil moisture volume that irrigation must maintain (if irrigation is active). It must lie between 0.0 and 1.0. As Land-use 1 is not irrigated we can set this to 0.

11 ***rdelay*** and ***mdelay*** define the time lag in days (including fractions of days) between water leaving the soil moisture storage and reaching the saturated zone (i.e. the aquifer). ***rdelay*** is usually larger than ***mdelay***, as the latter refers to fast-tracking macropore recharge. Both are functions of the depth to the saturated zone, and both are functions of soil permeability (especially ***rdelay***). Set these to as 5.0 and 1.0, respectively.

12 ***ks*** is the saturated hydraulic conductivity of the soil. Set this to 10.0.

13 ***m*** and ***l*** define the shape of the drainage rate vs stored water function and the pore connectivity, respectively. Set them both to 0.5. See the manual for explanations of these variables.

14 ***mflowmax*** defines the maximum volume of macropore recharge that can occur on any given day. Let us set it quite high at 0.1.

So far, your file should look something like this (we have used whitespace to separate values; you can use tab, whitespace or commas if you prefer):

```
* earth properties
0.2 0.0
5 1
10.0 1.0 0.5 0.1

* volume to elevation
offset factor1 factor2 power [elevmin elevmax]

* topographic surface
surface

* initial conditions
vol
nrbuf nmbuf
(rbuf(i), i=1,nrbuf)
(mbuf(i), i=1,nmbuf)

* solution parameters
nstep mxiter tol

* timing information
numdays noutdays
(outday(i),i=1,noutdays)

*data filenames
vegfile OR cropfac_all gamma_all
rainfile
epotfile
irrigfile OR irrigcode_all gwirrigfrac_all
```

15 Now we reach the **** volume to elevation*** and **** topographic surface*** sections of the LUMPREM input file. For this particular LUMPREM model, we are not concerned about these values (we will get to them later), but we still need to provide values for them in the LUMPREM input file. Provide the values that are shown below. These variables only affect how LUMPREM translates storage

volume to pseudo-head. They do not affect the storage calculations. As we are not interested in the pseudo-head from *lr_lu1*, the value of these variables is irrelevant.

```
* volume to elevation
0.0 2.0 3.0 0.5 -9999.0 10000.0

* topographic surface
0.0
```

16 In the **** initial conditions*** section of the LUMPREM input file we specify the starting volume of water in the soil moisture store, and the volume of any recharge that may be occupying the recharge buffers; presumably these were filled in days before the beginning of the simulation. The latter represents water which may have left the soil moisture store before the simulation started, but has not yet reached the saturated zone as recharge.

17 ***vol*** defines the starting volume of soil moisture. Let us assume that this is half of ***maxvol***. Replace "vol" with 0.1.

18 This next line gets a bit more complicated. ***nrbuf*** and ***nmbuf*** are the number of elements in the ***rbuf*** and ***mbuf*** delayed storage arrays (or lists). These arrays hold recharge that is on its way to the water table. The size of these arrays specifies the maximum time in days for which recharge can be delayed. The initial contents of these arrays hold recharge that is already on its way to the water table. Let us assume that there is no prior macropore recharge, but that two and three days before our simulation started, 0.002 m and 0.003 m left the soil moisture store, respectively. Let us further assume that the maximum number of days for which macropore recharge can be delayed is 1, while the maximum number of days for which normal recharge can be delayed is 3. Hence replace ***nrbuf*** and ***nmbuf*** with 3 and 1, respectively.

19 Now we need to record the initial contents of these ***rbuf*** and ***mbuf*** arrays. For ***rbuf*** there are three elements: 0.0, 0.002 and 0.003; for ***mbuf*** there is only one: 0.0. Note that the ordering of moisture stored in these arrays is 1 day before start, 2 days before start, and so on.

After you have made these replacements, the **** initial conditions*** section of the LUMPREM input file should look something like this:

```
* initial conditions
0.1
3 1
0.0 0.002 0.003
0.0
```

20 In the **** solution parameters*** section of the LUMPREM input file, set the values of ***nstep*, *mxiter*** and ***tol*** to 4, 500 and 1e-5, respectively. These specify the details of how LUMPREM iteratively solves the equation for stored soil moisture. If LUMPREM informs you that this iterative solution procedure has not converged, increase ***nstep*** and ***mxiter***. This will increase LUMPREM execution time; however, LUMPREM runs so quickly that this hardly matters. As a last resort you can increase ***tol***. You should not encounter convergence problems during this tutorial.

21 In the **** timing information*** section of the LUMPREM input file, LUMPREM's simulation time, and the times at which it must record information on its output file, are specified.

22 ***numdays*** defines the total number of simulation days. We are simulating two years. Set this to 731.

23 ***noutdays*** defines the number of instances at which LUMPREM-calculated quantities must be recorded on its output file. Suppose that we desire output at the end of each month of the 2-year

simulation. This means that there are 24 times at which information must be recorded. Replace ***noutdays*** with 24.

24 Next, we must inform LUMPREM of the days on which it must record outputs, by providing an array (list) of day numbers. LUMPREM will record the system status and accumulated fluxes at the end of these chosen days (at midnight). So, suppose that we wish to record outputs at the end of the first month of the simulation. The first month is January, which has 31 days. Therefore, the first entry in the array of output times must be 31 (the last day of the month).

25 Fill the ***outday*** list with day numbers that match the last day of each calendar month of the simulated period (use spreadsheet software such as EXCEL to help you). Note that you do not *have* to specify that LUMPREM record its outputs at the end of calendar months; in practice you can use any integer interval that you like (fractions of days are not accepted).

The **** timing information*** section of the LUMPREM input file should now look something like this:

```
* timing information
731 24
31 60 91 121 152 182 213 244 274 305 335 366 397 425 456 486 517 547 578
609 639 670 700 731
```

26 Finally, we reach the **** data filenames*** section of the LUMPREM input file. Here we will inform LUMPREM from which files it must obtain rainfall and potential evaporation time series. We will also set up how vegetation and irrigation variables behave over time.

27 Starting with the easy ones, replace ***rainfile*** and ***epotfile*** with the respective filenames. In our case these are named *rain.dat* and *epot.dat*.

28 How we define the ***vegfile*** and ***irrigfile*** variables depends on how we want these parameters to vary over time. If we want them to be constant over the entire simulation, we can simply provide their values directly in the LUMPREM input file. Alternatively, we can create additional files with time-varying values. For now, let us assume constant values (we will use time-varying values later in the tutorial).

29 For vegetation, let us assume that crop factor and gamma are both 1.0 over the entire simulation time.

30 As stated above, Land-use 1 (the land-use for which the current instance of LUMPREM is being set up) is not irrigated, so we can set ***irrigcode*** to 0 (0 means off; 1 means on). As there is no irrigation, ***gwirrigfrac*** becomes irrelevant; however, it still requires a value. Set it to 0.0.

The **** data filenames*** section of the LUMPREM input file should now look something like this:

```
* data filenames
1.0 1.0
rain.dat
epot.dat
0 0.0
```

OK, great! We are now ready to run our first LUMPREM model.

31 Open a command line window in your working folder and type "*lumprem lr_lu1.in lr_lu1.out*" at the prompt. This command tells LUMPREM to run using the *lr_lu1.in* input file and to write results to file *lr_lu1.out*. Press <enter>.

32 Alternatively, you can use a batch file to automate the process of responding to user prompts. An example batch file (*run_models.bat*) is provided in the *tutorial_1* folder.

33 If all goes well, you should see something like this on your screen:

```
C:\..\your working folder > lumprem lr_lu1.in lr_lu1.out
 - file lr_lu1.in read ok.
 - file lr_lu1.out written ok.
```

Check your working folder. You should find a new file called *lr_lu1.out*. Open it and take a look. There are many columns (see the manual for descriptions of each of them). Copy (or open) the file into your preferred spreadsheet or graphing package to become familiar with the outputs.

2.3 Setting up an *irrigation* file

Recall from the start of this tutorial that our model domain includes areas with three different land-uses. We have just set up a LUMPREM model for Land-use 1, which has no irrigation. Land-use 2 and Land-use 3 are irrigated. Land-use 2 has year-round irrigation. Land-use 3 is only irrigated during half of the year. For simplicity, let us assume that all other soil and vegetation characteristics are the same as for Land-use 1.

34 Make two copies of *lr_lu1.in*. Call them *lr_lu2.in* and *lr_lu3.in* respectively.

35 In both of these files, set ***irrigvolfrac*** to 0.2 (see point 10 above, as well as the LUMPREM manual).

36 In *lr_lu2.in*, change the values ascribed to the ***irrigcode*** and ***gwirrigfrac*** variables (see point 30 above)**.** Irrigation is active, so set ***irrigcode*** to 1 (recall that 1 means irrigation is on, whilst 0 means it is off). At our hypothetical site, all irrigation is supplied from groundwater, so set the fraction of irrigation supplied by groundwater (***gwirrigfrac***) to 1.0.

The last section of file *lr_lu2.in* should now look something like this:

```
* data filenames
1.0 1.0
rain.dat
epot.dat
1 1.0
```

37 Run LUMPREM using *lr_lu2.in* as its input file and save its outputs in file *lr_lu2.out* (see point 31).

38 Now, for *lr_lu3.in* we want to make irrigation vary over time. So, we need to instruct LUMPREM to read an external file that contains an irrigation time series (which we must create). Let us start by writing this file.

39 The irrigation file must have three columns which contain the ***day***, ***irrigcode*** and ***gwirrigfrac*** variables. The columns specify the days on which irrigation begins/ends and the fraction of irrigation that is supplied by groundwater. For simplicity, we will assume that the latter is always 1.0.

40 In Land-use 3, we happen to know that farmers commence irrigation on the 1st of May and cease irrigation on the 30th of September every year (conveniently). However, the first line of the irrigation file must always refer to day 1. As the simulation starts with no irrigation, set ***irrigcode*** and ***gwirrigfrac*** to 0.0 on the time-period which begins on day 1.

41 As our simulation starts on the 1st of January 2000, irrigation will begin on simulation day 122. Irrigation ceases on day 274. And then starts again in the following year on day 487. And so on. Write the remaining lines into the irrigation file and save it as *irrig.dat*. It can be saved as whitespace or comma delimited values. We have used whitespace. It should look something like this:

```
1      0      0
122    1      1
275    0      0
487    1      1
640    0      0
```

42 Run LUMPREM with *lr_lu3.in* as its input file. Save its outputs to file *lr_lu3.out* (see point 31).

43 Let's take a look at the results and see how Land-use2 and 3 differ. Our outputs are shown in the plot below. Note how irrigation continues year-round for Land-use2, whilst irrigation for Land-use3 drops to zero during half the year.

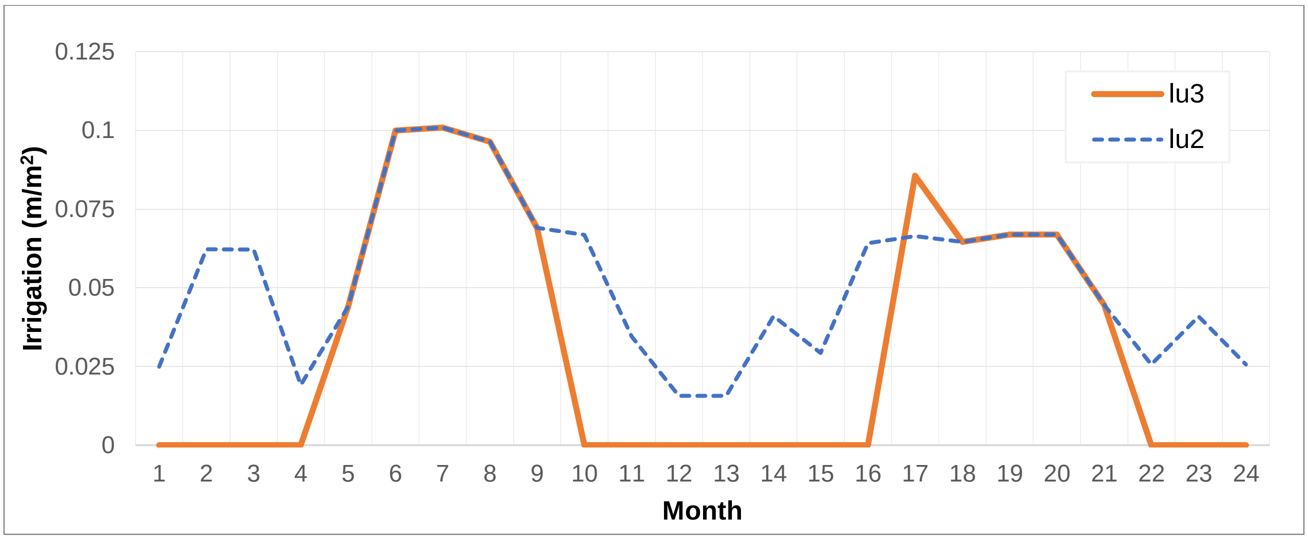

Figure 3 LUMPREM simulated irrigation volumes for Land-use 2 and Land-use 3.

2.4 Defining *elevation* or pseudo-head

LUMPREM records the daily volume of stored water in its output file. This quantity can rise steeply after rainfall and decay slowly thereafter. As such, its behaviour is not unlike that of groundwater levels in a shallow aquifer. LUMPREM allows a user to calculate a "pseudo groundwater head" time series from daily volumes using a nonlinear function whose parameters can be set by the user (and/or adjusted through history matching). This pseudo-head time series is also recorded in LUMPREM's output file. Like other time series that are recorded in this file, it can serve as an input time series for a groundwater model.

We wish to assign a time varying elevation to the GHB that forms one of the boundaries of our groundwater model. We shall now set up a LUMPREM model to generate this elevation time series.

Make another copy of *lr_lu1.in.* Call it *lr_ghb.in*. For simplicity we will keep all parameters the same, except for those which reside in the **** volume to elevation*** and **** topographic surface*** sections.

The **** volume to elevation*** section houses six LUMPREM variables. See the LUMPREM manual for the equation that these variables inform. In brief, ***offset***, as the name suggests, provides an offset from the storage volume to some elevation. The next three variables (***factor1***, ***factor2*** and ***power***) deform the storage time series so that it looks more like an elevation time series. Lastly, ***elevmin*** and ***elevmax*** set bounds for the elevations that are calculated by this equation. As their names suggest, these represent minimum and maximum possible groundwater levels.

Change the values for these variables in file *lm_ghb.in* to: 50.0, 10.0, 10.0, 0.1, -9999.0 and 10000.0 respectively. We are simply setting the maximum and minimum elevation as extremely high and extremely low, respectively. This way they will not clip LUMPREM's outputs.

The ***surface*** variable in the **** topographic surface*** section of the LUMPREM input file defines the elevation of the topographic surface. This will be used to set the depth to groundwater calculated by LUMPREM. In the present instance, we set it to 80.0 metres.

Run LUMPREM and take a look at the elevation time series which it calculates. If you plot LUMPREM-calculated elevation of *lr_lu1* and *lr_ghb* you can see how the variables discussed above transform LUMPREM's output. You should see something like the graph below. Note that the scale for *lr_ghb* elevation is plotted on the left axis of this graph while and the scale for elevation of *lr_lu1* is shown on the right-axis. Note how the elevation of *lr_ghb* is offset by more than 50m and has a larger variability than of *lr_lu1*. Recall that the offset in *lr_lu1* is zero; hence elevation is similar to the simulated values of storage volume.

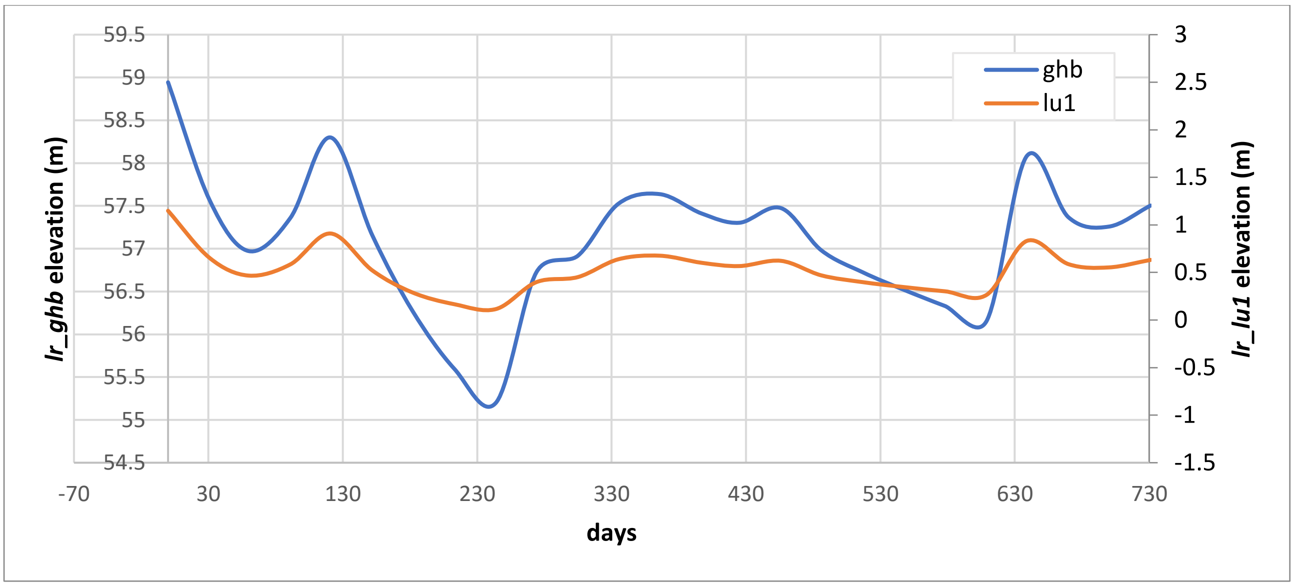

Figure 4 LUMPREM simulated elevation time series from the *lr_lu1* and *lr_ghb* models.

If you like, play around with the values for ***offset, factor1, factor2*** and ***power*** to get a feel for their impact. However, in practice these would usually be adjusted by PEST or PEST++ during history matching of the composite MODFLOW 6/LUMPREM model so that the outputs of this model match field observations.

Excellent, so we now have several LUMPREM models set up and running. The next step is to generate MODFLOW 6 time series input file (i.e.TS6 files) from LUMPREM outputs. We will address this in Chapter 3.

The *tutorial_1* folder contains all files generated up to this point, as well as a batch file to run each of the LUMPREM models sequentially.

3. FROM LUMPREM TO MODFLOW 6

In Chapter 2 of this document we described the process of setting up and parameterizing several LUMPREM models. The purpose of these LUMPREM models is to generate inputs for a groundwater flow model. The current chapter demonstrates how to convert LUMPREM-generated outputs into files which can be read by MODFLOW 6 stress packages. This chapter is divided into two parts. The first part demonstrates the use of the LR2SERIES utility to generate MODFLOW 6 time series files (TS6 files) from LUMPREM output files. The second part demonstrate the use of TS6PROC to apply additional transformations to these TS6 files.

As previously described, the underlying purpose of this workflow is to facilitate data assimilation in order to quantify and constrain the uncertainties of a groundwater model's predictions. To this end all of the steps described herein need to be carried out as part of an automated process which can be called by PEST/PEST++. The following chapter will describe how to set up the steps of an automated workflow which converts LUMPREM outputs into MODFLOW 6 inputs. However, it will not cover set up of PEST/PEST++.

The workflow demonstrated herein is specific to MODFLOW 6. It is assumed that the reader is familiar with MODFLOW 6 and its file structures. That being said, the concepts are directly translatable to other modelling codes. For example, actions carried out by LR2SERIES and TS6PROC can easily be replicated using FEFLOW's embedded Python interface or plugin functionality.

3.1 Using LR2SERIES

We are now going to use the LR2SERIES utility to convert LUMPREM outputs into MODFLOW 6 time series files. "LR2SERIES" stands for "LUMPREM output file to time series file". This utility reads one or a number of LUMPREM output files and writes one or a number of MODFLOW 6 time series files. The suite of time series that it writes to the latter files are the same as, or modified from, those read from LUMPREM output files.

LR2SERIES receives instructions on what it must do from a control file supplied by the user. We will now construct that file. There are two actions that LR2SERIES takes. (1) It reads one or a number of LUMPREM output files, and (2) it writes one or a number of MODFLOW 6 time series files. A single LR2SERIES control file can instruct LR2SERIES to read multiple LUMPREM outputs (resident on multiple LUMPREM output files), and write multiple time series to multiple MODFLOW 6 time series files.

If you are starting fresh, the *tutorial_1* folder contains all files generated up to this point. Alternatively, if you have completed Chapter 2 of this tutorial, you can continue using the files you have generated.

44 Open a new blank text file in your text editor of choice. Save it as *lr2series.in*.

45 We will start by informing LR2SERIES of the LUMPREM outputs that we want it to read. A line with the text "READ_LUMPREM_OUTPUT_FILE" instigates the action to read a file. It must be followed by the name of the file that must be read, and the number of columns that must be read from that file.

46 Let us start with the output from *lr_lu1*. From this LUMPREM model's output file, we only need the time series that reside in the ***recharge*** and ***gw_pot_evap*** columns. Type in the following. Note the read file command, followed by the name of the LUMPREM output file, followed by the number 2. This command informs LR2SERIES that it must read two columns from the LUMPREM output file *lr_lu1.out*.

```
READ_LUMPREM_OUTPUT_FILE      lr_lu1.out  2
```

47 Following this header, the next two lines of the LR2SERIES input file must inform LR2SERIES which columns to read, and how to read them. Each of these subsequent lines must contain three entries, these being as follows:

- A user-supplied name for the time series that will be recorded on the MODFLOW 6 time series file (and then used by MODFLOW 6 itself). Note that each name supplied for a MODFLOW 6 time series must be unique.
- The name of the LUMPREM output file column to read.
- Whether to divide LUMPREM-generated time series values by the time-interval over which they were accumulated. These two options are distinguished through use of either the *div_delta_t* or *no_div_delta_t* string.

48 Let us start by reading the ***recharge*** column from the LUMPREM output file. Name the corresponding MODFLOW 6 time series as *rch_lu1*. Recall from LUMPREM documentation that LUMPREMs output for recharge is the accumulated recharge volume over the time-interval that prevails since the last output time. However, MODFLOW 6 expects a rate. Hence, we must divide the LUMPREM-generated quantity by the time-interval over which it was accumulated. To do this, simply provide LR2SERIES with the instruction *div_delta_t*.

49 Do the same for the ***gw_pot_evap*** column of the LUMPREM output file. The pertinent section of the LR2SERIES control file should look something like this:

```
READ_LUMPREM_OUTPUT_FILE      lr_lu1.out  2
rch_lu1           recharge          div_delta_t
evp_lu1           gw_pot_evap       div_delta_t
```

50 OK, great. We can now progress to other LUMPREM output files. You can repeat the same procedure as above for Land-use 2 and Land-use 3. Just copy and paste this section of the LR2SERIES input file to create another section; alter the name of the LUMPREM output file that must be read, and provide a new name for the MODFLOW 6 time series. We recommend using the same naming convention as that which has been employed up until now, namely *rch_lu2*, *rch_lu3,* etc.

51 There is one further change necessary for Land-use 2 and Land-use 3. Recall that we wish to extract groundwater for irrigation in these land-use areas. Therefore, we need to provide time series to the WEL package to simulate abstraction. The LUMPREM output column ***gw_withdrawal*** provides simulated abstraction volumes. So, we need to read an additional column from the LUMPREM output file.

52 For Land-uses 2 and 3, the number of LUMPREM columns that must be read should be altered from 2 to 3; this requires the addition of a new line. Name the MODFLOW 6 well time series *wel_lu2* and *wel_lu3,* respectively.

53 Finally, we need to read the ***elevation*** column from *lr_ghb.out*. Name the respective MODFLOW 6 time series *ghb*.

54 Note that the ***elevation*** output column that is recorded on the LUMPREM output file records elevations at discrete points in time. Therefore, respective MODFLOW 6 time series do not need to be divided by the length of the time interval that has elapsed since the last LUMPREM output time. This is denoted using the text string "*no_div_delta_t*".

55 By now your LR2SERIES input file should look something like this:

```
READ_LUMPREM_OUTPUT_FILE      lr_lu1.out  2
rch_lu1           recharge          div_delta_t
evp_lu1           gw_pot_evap       div_delta_t

READ_LUMPREM_OUTPUT_FILE      lr_lu2.out  3
rch_lu2           recharge          div_delta_t
evp_lu2           gw_pot_evap       div_delta_t
wel_lu2           gw_withdrawal     div_delta_t

READ_LUMPREM_OUTPUT_FILE      lr_lu3.out  3
rch_lu3           recharge          div_delta_t
evp_lu3           gw_pot_evap       div_delta_t
wel_lu3           gw_withdrawal     div_delta_t

READ_LUMPREM_OUTPUT_FILE      lr_ghb.out  1
ghb               elevation         no_div_delta_t
```

56 We have now informed LR2SERIES what to read. Next, we need to tell it what to do with the time series which it has just read. This is accomplished using one or a number of “WRITE_MF6_TIME_SERIES_FILE” blocks. Each command of this type must also be followed by the name of the MODFLOW 6 time series file that must be written, and by the number of time series that must be recorded on that file. Optionally this must be followed by a number which specifies a time series shift; see LR2SERIES documentation for further details.

57 Our MODFLOW 6 model has two stress periods: it starts with a steady-state stress-period; this is followed by a transient stress-period. LUMPREM calculates MODFLOW 6 inputs for the transient stress-period. (The fact that these inputs can change over the time spanned by a single stress period distinguishes MODFLOW 6 from previous versions of MODFLOW. This is one of the outcomes of using time series to express these changing stresses.) Introduction of a steady state stress period to precede the time-varying stresses creates a time-keeping problem as LUMPREM simulation time will not match MODFLOW 6 simulation time after the introduction of this steady state stress period. This time series mismatch will occur because LUMPREM day *zero* will correspond to MODFLOW 6 day *one*; the preceding steady state stress period will be of one day’s duration So we need to introduce this time offset when building a MODFLOW 6 time series file from one or a number of LUMPREM output files. At the same time, values must be supplied for steady state stresses (for example, recharge and evapotranspiration) which precede transient stresses. This matter will be addressed later in this tutorial when use of the TS6PROC utility is explained.

58 Let us start with the time series file that will be supplied for the recharge package. Let us call this file *rch.ts*. Recall that we have employed three LUMPREM models to calculate three recharges time series - one for each Land-use. So, we must inform LR2SERIES of this by recording the number “3” on the header to this block. Then follows the number “1.0”. As was described above, this informs LRSERIES that LUMPREM-calculated time series must be delayed by a day before being provided to MODFLOW. The header for the block should look something like this:

```
WRITE_MF6_TIME_SERIES_FILE    rch.ts      3     1.0
```

59 The next three lines must identify which time series to write to the MODFLOW 6 time series file. They must also record the properties that MODFLOW 6 requires for each time series, namely ***scale***, ***offset*** and ***method***; see documentation of MODFLOW 6 for the roles of these variables.

60 Three Land-use *rch* time series must be recorded in file *rch.ts*. For each of these we will assign values of 1, 0.0 and *linearend* to the MODFLOW 6 ***scale***, ***offset*** and ***method*** variables, respectively.

61 Recall that time series supplied to MODFLOW 6 were delayed by a day. LR2SERIES introduces a new element to each time series that pertains to time zero. Values are needed that correspond to this time in each time series. LR2SERIES provides two options for this; either a value can be provided, or the text "next" can be provided. The later instructs LR2SERIES to use the value associated with the first non-zero time series element; this is the element that used to be the first of the time series prior to time series delay and insertion of the new time series element. In our case it does not matter what value we will use, as time series values pertaining to a time of zero will be replaced shortly. Let us therefore use "next" for all time series. Feel free to experiment. This block of the LR2SERIES file should look something like this:

```
WRITE_MF6_TIME_SERIES_FILE    rch.ts      3     1.0
rch_lu1          1           0           linearend   next
rch_lu2          1           0           linearend   next
rch_lu3          1           0           linearend   next
```

62 Repeat this process for the EVT, WEL and GHB TS6 files. Call the TS6 files *evp.ts*, *wel.ts* and *ghb.*ts, respectively. The GHB time series method should be set to *linear* instead of *linearend*.

63 If your file looks like this, you are ready to run LR2SERIES:

```
READ_LUMPREM_OUTPUT_FILE      lr_lu1.out  2
rch_lu1          recharge          div_delta_t
evp_lu1          gw_pot_evap       div_delta_t

READ_LUMPREM_OUTPUT_FILE      lr_lu2.out  3
rch_lu2          recharge          div_delta_t
evp_lu2          gw_pot_evap       div_delta_t
wel_lu2          gw_withdrawal     div_delta_t

READ_LUMPREM_OUTPUT_FILE      lr_lu3.out  3
rch_lu3          recharge          div_delta_t
evp_lu3          gw_pot_evap       div_delta_t
wel_lu3          gw_withdrawal     div_delta_t

READ_LUMPREM_OUTPUT_FILE      lr_ghb.out  1
ghb              elevation         no_div_delta_t


WRITE_MF6_TIME_SERIES_FILE    rch.ts            3     1.0
rch_lu1          1     0     linearend   next
rch_lu2          1     0     linearend   next
rch_lu3          1     0     linearend   next

WRITE_MF6_TIME_SERIES_FILE    evp.ts            3     1.0
evp_lu1          1     0     linearend   next
evp_lu2          1     0     linearend   next
evp_lu3          1     0     linearend   next

WRITE_MF6_TIME_SERIES_FILE    wel.ts            2     1.0
wel_lu2          1     0     linearend   next
wel_lu3          1     0     linearend   next

WRITE_MF6_TIME_SERIES_FILE    ghb.ts            1     1.0
ghb              1     0     linear      next
```

64 Open the command prompt, type *lr2series* and press <enter>.

65 You will be prompted for the name of its input file. Type *lr2series.in* and press <enter>.

66 Alternatively, you can use a batch file to automate the process of responding to user prompts. An example batch file (*run_lr2series.bat*) is provided in the *tutorial_2* folder.

67 If everything is set up correctly you should see something like the following on your screen:

```
C:\..\your working folder >lr2series
 Enter name of LR2SERIES control file: lr2series.in

 - reading file lr2series.in...

 - reading LUMPREM2 output file lr_lu1.out...
 - file lr_lu1.out read ok.
 - reading LUMPREM2 output file lr_lu2.out...
 - file lr_lu2.out read ok.
 - reading LUMPREM2 output file lr_lu3.out...
 - file lr_lu3.out read ok.
 - reading LUMPREM2 output file lr_ghb.out...
 - file lr_ghb.out read ok.
 - writing file rch.ts...
 - file rch.ts written ok.
 - writing file evp.ts...
 - file evp.ts written ok.
 - writing file wel.ts...
 - file wel.ts written ok.
 - writing file ghb.ts...
 - file ghb.ts written ok.

 - file lr2series.in read ok.
```

68 Open up the TS6 files that LR2SERIES has just written and take a look. These can now be used in a MODFLOW 6 model.

69 Great job! However, for the particular MODFLOW 6 model that we want to build, one additional step is required: inserting values for the first steady-state stress-period. This will be covered in Chapter 3.2.

70 The folder *tutorial_2* contains all files generated up to this point as well as an additional batch file to run LR2SERIES.

3.2 Manipulating Time Series files with TS6PROC

At this stage we have time-varying inputs for our various packages during the transient part of our simulation. But, as you recall, our model is set up to begin with an initial steady-state stress period; this is followed by a transient stress-period. It is not uncommon for transient models to begin with a steady-state stress period; this provides the opportunity to get initial heads approximately right for the ensuing transient run.

Let us assume that the time-averaged value of each of our input time series is representative of the steady-state inputs of the same type. Conceptually, we could just calculate these average values, manually assign them to the first stress period of our MODFLOW 6 model and carry on. However, we are sophisticated people and want to estimate LUMPREM parameters together with MODFLOW 6 parameters when we calibrate our joint model (comprised of three LUMPREM models and MODFLOW 6); we also want to analyse the uncertainties of these parameters. So, we need to be able to automatically update average values of LUMPREM model outputs on each occasion that the values of LUMPREM parameters are altered by PEST/PEST++. For this we can use TS6PROC, a utility program included as part of the PEST software suite.

TS6PROC was written specifically to manipulate MODFLOW 6 time series when a model is run under the control of PEST or PEST++. TS6PROC reads a MODFLOW 6 time series file (referred to as a TS6 file from now on). It manipulates time series that are contained in this file in ways that are specified through a TS6PROC input control file. It then writes a new TS6 file that contains the altered time series. This altered file can be used by MODFLOW 6 in place of the original one.

A note on terminology for clarity: a TS6 “time series file” is a file read by MODFLOW 6 which can contain one or several time series. For example, in the previous chapter we created a TS6 file called *wel.ts*. That TS6 file contains two time series; these are named *wel_lu2* and *wel_lu3,* respectively. See documentation of MODFLOW 6 for full details.

TS6PROC offers a number of options for construction and manipulation of time series. A time series can be built and/or modified through application of equations of arbitrary complexity that apply to all terms of one or more existing series. These equations can feature parameters that can be manipulated by members of the PEST and/or PEST++ suites when undertaking model calibration and/or calibration-constrained uncertainty analysis. TS6PROC also provides a number of special functions for manipulation of time series. In our case, we are only going to make use of one of these functions. Other tutorials may explore TS6PROC in greater detail, thereby demonstrating the use of other TS6PROC functions.

TS6PROC requires that the user prepare an input file which tells it what to do. This file contains three blocks which outline: (1) the files that it must read and write, (2) the parameters to use in functions that create or alter time series, and (3) the processes that it must carry out. The reader is advised to read TS6PROC documentation for details of each of these. In this document we only provide details that are pertinent to our current workflow.

If you are starting afresh, the *tutorial_2* folder contains all files generated up to this point. Alternatively, if you have completed Chapter 3.1, you can continue using the files that you have generated.

71 Our use of TS6PROC in this case is rather simple. For each of the time series created above from LUMPREM outputs, we wish to replace the value for day 0 with the time averaged value of the rest of the time series (recall that we shifted each time series backwards to make room for this initial value). Luckily, our friendly software developers have included functions to do just this: the *time_average_over_interval()* and *assign_terms()* functions.

72 Let us begin by setting up a TS6PROC input file that reads and adjust time series contained in the *well.ts* TS6 file. Open a new file in a text editor and type in the text below:

```
BEGIN FILES
 FILEIN wel.ts
 FILEOUT wel_new.ts
END FILES

BEGIN PARAMETERS
END PARAMETERS
```

73 This tells TS6PROC to read file *wel.ts* (which we created using LR2SERIES in the previous chapter of this tutorial) and to write a new file named *wel_new.ts* after it has finished processing the time series contained in the former file. *wel_new.ts* will become the TS6 file that MODFLOW 6 actually reads.

74 We have left the PARAMETER block empty for now, as we are not using any parameters in this case. Hence there are no lines between BEGIN PARAMETERS and END PARAMETERS.

75 Next, we need to fill out the PROCESSING block of the TS6PROC input file. This is where we define what operations to apply to the time series that were read.

76 First, let us obtain the time averaged value of each of the time series contained in file *wel.ts* from day 1 onwards. Recall that the TS6 file *wel.ts* has two time series, and that these are named *wel_lu2* and *wel_lu3*.

77 We will employ the *time_average_over_interval()* function on each time series and assign each of the averages that it calculates to new variables. This function takes four arguments: (1) the time series name, (2) whether log transformation is required before averaging, (3) the beginning of the time interval over which averaging takes place and (4) the end of the time interval over which averaging takes place. Starting with time series *wel_lu2*, let us create a new variable named *avg2* to which we assign the time average of time series elements. To do this, simply type in the name of the function, the values of its arguments, and the name of the new variable to which the average is assigned as in the example below:

```
BEGIN PROCESSING
 avg2 = time_average_over_interval(wel_lu2, none, 1.0, 99999.0)
END PROCESSING
```

78 We defined *avg2* as the average of values from time series *wel_lu2* between time 1.0 and time 99999.0 (the latter is simply a very high number which goes beyond the end of the time series). The argument *none* specifies that the function must average the actual values of time series elements rather than their logs.

79 Now we need to tell TS6PROC what to do with the new quantity that it has just created. Recall that we want to replace the value at time zero in our original time series with this average. For this we will use the *assign_terms()* function to alter the *wel_lu2* time series.

80 TS6PROC syntax demands that every function assign a value to something. This can be another time series; or it can be a scalar quantity that is associated with a single value. By typing in the variable name *wel_lu2* followed by an equal sign followed by the name of the function (of which *wel_lu2* is one of its argument), we are informing TS6PROC that it must update the *wel_lu2* time series according to the function. Type in the text shown below:

```
BEGIN PROCESSING
 avg2 = time_average_over_interval(wel_lu2, none, 1.0, 99999.0)
 wel_lu2 = assign_terms(wel_lu2, avg2, 0.0, 0.0)
END PROCESSING
```

81 Note that we have updated the time series *wel_lu2*, by assigning the value *avg2* to time elements between 0.0 and 0.0. In our case, the latter could be any value equal or greater than 0.0 or lower than (but not equal to!) 1.0, the time for which our transient period begins.

82 Save the file as *ts6wel.in*. Now let us run TS6PROC with this file as the input file, so that we can see the changes that we have implemented.

83 Open the command line, type *ts6proc ts6wel.in* and press <enter>.

84 Alternatively, you can use a batch file to automate the process of responding to user prompts. An example batch file (*run_ts6proc.bat*) is provided in the *tutorial_3* folder.

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

```
C:\..\your working folder >ts6proc ts6wel.in

 TS6PROC Version 1.00. Watermark Numerical Computing.

 - reading file wel.ts...
 - file wel.ts read ok.

 Processing: average2 = time_average_over_interval...
 Processing: wel_lu2 = assign_terms(wel_lu2, avera...

 - writing file wel_new.ts...
 - file wel_new.ts written ok.
```

86 Open up *wel.ts* and *wel_new.ts* in a text editor and compare them. Note how the *wel_lu2* time series in *wel_new.ts* has changed in accordance with the instructions that we provided above.

87 Great. So now we can do the same for the *wel_lu3* time series. Update the TS6PROC *ts6wel.in* input file with functions to alter the *wel_lu3* time series in addition to the *wel_lu2* time series. The revised file should look something like this:

```
BEGIN FILES
 FILEIN wel.ts
 FILEOUT wel_new.ts
END FILES

BEGIN PARAMETERS
END PARAMETERS

BEGIN PROCESSING
 avg2 = time_average_over_interval(wel_lu2, none, 1.0, 99999.0)
 avg3 = time_average_over_interval(wel_lu3, none, 1.0, 99999.0)

 wel_lu2 = assign_terms(wel_lu2, avg2, 0.0, 0.0)
 wel_lu3 = assign_terms(wel_lu3, avg3, 0.0, 0.0)

END PROCESSING
```

88 Run TS6PROC again with the *ts6wel.in* input file and confirm your outputs. If you are content with your results, you can now proceed to do the same for each of the other TS6 files which we previously generated using LR2SERIES (*rch.ts, evp.ts* and *ghb.ts*). Note that you will need to create an individual TS6PROC input file to process each of these TS6 time series files.

89 Congratulations, you now have a workflow to generate time-varying MODFLOW 6 inputs from several LUMPREM models. All of which can be integrated into a PEST/PEST++ history-matching or uncertainty analysis process.

90 The *tutorial_3* folder contains all files generated up to this point. Individual batch files (**.bat*) for each of the steps in the workflow have been provided in this folder, along with a single batch file which runs all the steps in one go (*run_all.bat*). In practice, PEST/PEST++ might call *run_all.bat* after adjusting parameters within the LUMPREM input files. More complicated setups might also consider adjustments to variables in TS6PROC input files. Although possible, it is unlikely that LR2SERIES input files would be included as parameters in a PEST/PEST++ process.

gmdsi

Groundwater Modelling Decision Support Initiative

gmdsi.org

BHP

Flinders UNIVERSITY

NATIONAL CENTRE FOR GROUNDWATER RESEARCH AND TRAINING

RioTinto