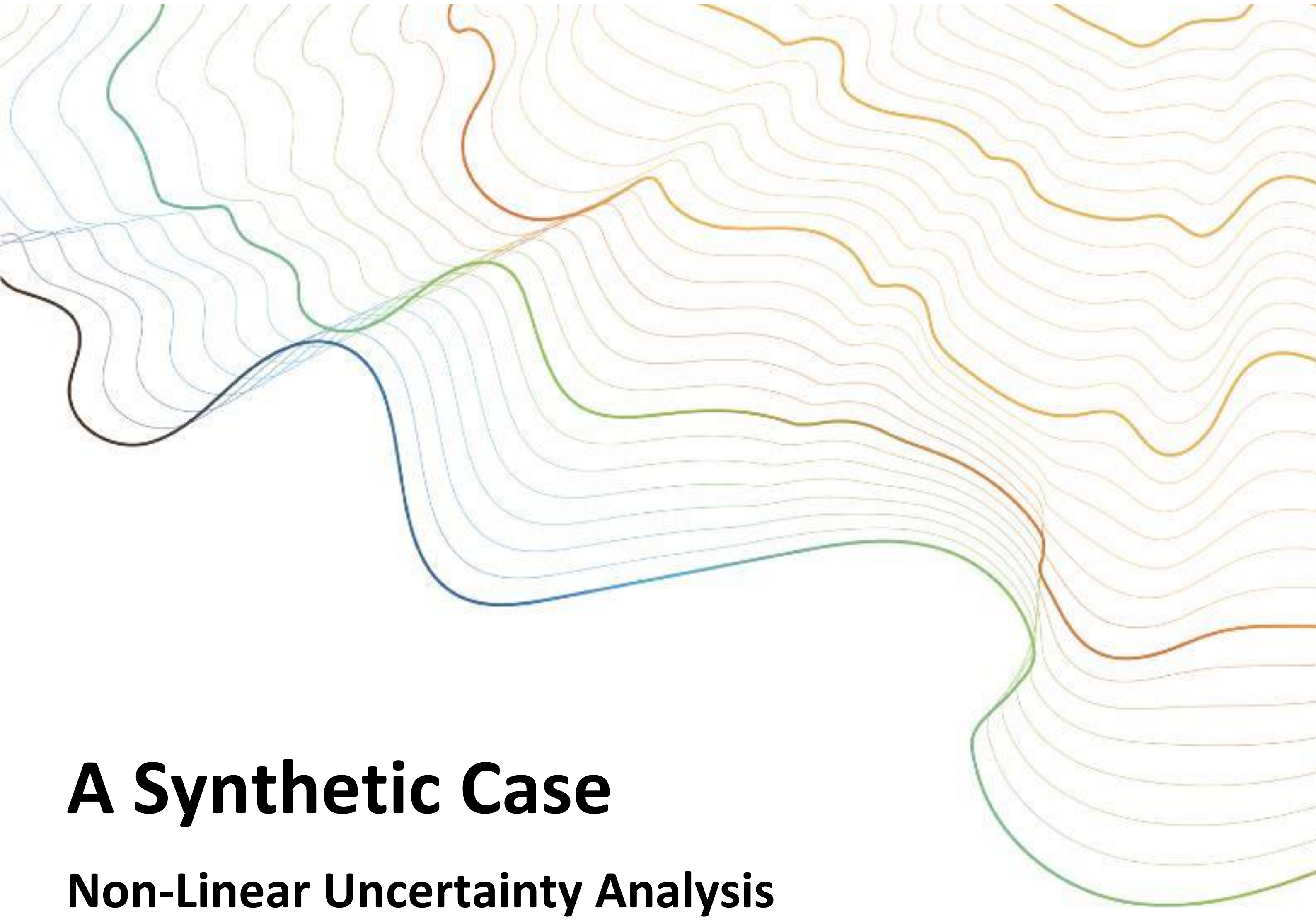

A Synthetic Case

Non-Linear Uncertainty Analysis

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

PREFACE

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

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

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

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

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

Rui Hugman

John Doherty

CONTENTS

1. INTRODUCTION

This document is part of series of tutorials which demonstrate workflows for parameter estimation and uncertainty analysis with PEST/PEST++. These are not the only (or necessarily the best) workflows; their purpose is to take the reader through the basics of how to accomplish common tasks. The present document is a tutorial on how to perform common non-linear uncertainty analysis tasks using PESTPP-IES (henceforth referred to as IES). These tasks are applied to a model which has been setup and calibrated in a previous GMDSI tutorial It will be the subject of linear analysis in subsequent tutorials.

The PEST roadmaps are a useful complement to this series of tutorials. The current tutorial is complemented by **Roadmap 13: Nonlinear Posterior Uncertainty Analysis** (henceforth referred to as “the Roadmap”). The Roadmap provides a brief discussion of the concepts and theory, as well as an outline of steps required to undertake non-linear predictive uncertainty analysis. The tutorial focuses on practical implementation of concepts discussed in the Roadmap. The aforementioned GMDSI tutorial on model calibration provides details of the model that is used in the present tutorial. It also provides the PEST dataset configuration on which the current tutorial is based. A summary is provided in the following section.

This tutorial assumes that you are familiar with PEST in particular, and the PEST software suite in general. Although completion previous tutorials is not a required precursor to completion of the present tutorial, it is recommended that you do so if you are not familiar with PEST. Detailed descriptions of how PEST++ differs from PEST, or of the intricacies of the various options available for PEST++ configuration are beyond the scope of this tutorial. In the tutorial *documentation* folder, along with the current document you will find two additional files. These are the PEST++ v5 user guide (*pestpp_users_guide_v5.0.11.docx*) and the USGS documentation of the code (*tm7c26.pdf*). The former provides detailed descriptions of what programs belonging to the PEST++ suite do, as well as suggestions for their use. The later provides an easy-to-read overview of PEST++ with an example workflow; however it assumes a certain level of familiarity with the software. Up-to-date versions of these documents can be found here and here, respectively.

Completed versions of all files generated during the tutorial can be found in the folder named *completed*. These are useful if you need to troubleshoot your own files, or if you wish to jump into the tutorial at a specific point. That being said, it is recommended that you at least read documentation for those parts of the tutorial that you do not complete yourself.

Software Executables

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

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

1.1 Background

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

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

The nearby town of Makebelievesville needs to expand its water supply through groundwater abstraction. Land access and existing infrastructure limit possible sites for a wellfield. Of these, the preferred site is close to one of the streams; see the red star in Figure 1. Proposed abstraction rates are a constant 2,000 m^3/d. Proximity to the stream raises concerns that streamflow may be depleted. After extensive research, ecologists have determined that the frog species requires a minimum flow rate of 6,500 m^3/d to survive. If proposed abstraction reduces groundwater discharge to the streams to less than this value, "a bad thing" happens.

A numerical model was developed to assess whether proposed abstraction will reduce groundwater discharge to the streams below the critical limit of 6,500 m^3/d under long-term average conditions (i.e. steady state). This model simulates two steady state stress periods. The first represents historic conditions. The second accounts for the proposed additional abstraction.

Model parameters were calibrated against hydraulic head and stream flow measurements. Recharge rates, hydraulic conductivities in each model layer, and conductances of boundary conditions were parameterized using pilot points. Preferred-value Tikhonov regularisation was implemented; this was accompanied by the use of covariance matrices to constrain patterns of emergent heterogeneity. Linear uncertainty analysis was undertaken following model calibration in order to approximately characterise parameter and predictive uncertainty, and to explore the worth of collecting additional observations.

The model prediction of interest is groundwater flow into the upper stream reach during the second stress period; this observation is named *p-trib_1* in the PEST control file.

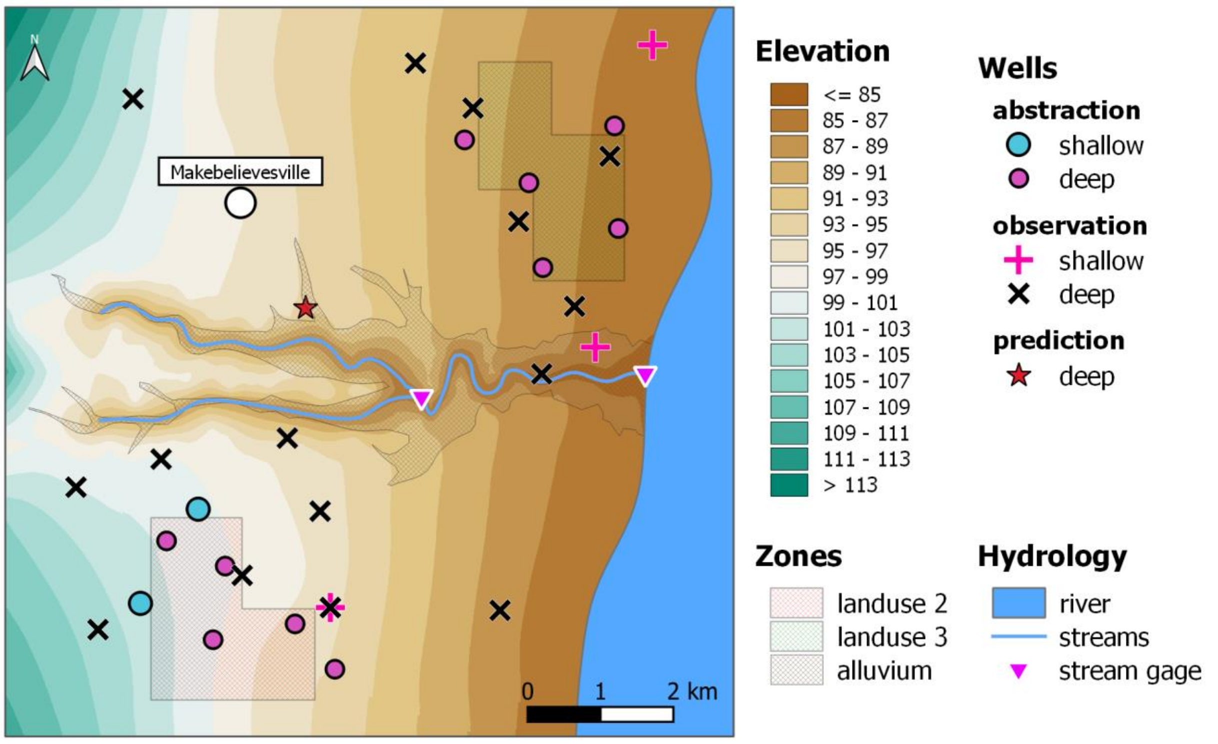

Figure 1 – Layout of the synthetic case and main features of interest.

2. NON-LINEAR UNCERTAINTY ANALYSIS

In contrast to linear uncertainty analysis, non-linear methods do not suffer from the limitation of assuming a linear relationship between model predictions and model parameters. Given that such relationships are rarely linear, this is an important consideration. Nevertheless, non-linear methods are not free of assumptions. These include those associated with the conceptual model on which the numerical model is based, and those associated with prior parameter probability distributions. Of particular importance is the potential for high permeabilities to exhibit different patterns from those exhibited by low permeabilities, and for regions of high permeability to exhibit greater connectedness than regions of low permeability. Unfortunately, the effects of this and other types of geologically "realistic" hydraulic property dispositions are difficult to include in either linear or non-linear uncertainty analysis.

Non-linear methods can sometimes incur a higher implementation cost than linear methods, and do not facilitate the calculation of value-added quantities such as data-worth. That being said, under some circumstances, non-linear uncertainty analysis based on the IES iterative ensemble smoother can be numerically cheaper to implement than linear uncertainty analysis, as we will see later in the tutorial.

IES sample the posterior parameter probability distribution. It commences by using a suite of random parameter fields sampled from the prior parameter probability distribution. Each parameter field is referred to as a "realisation". The suite of realisations is referred to as an "ensemble". Using an iterative procedure, IES modifies the parameters that comprise each realisation, so that each is better able to better reflect historical data. In other words, IES adjusts the parameters of each realisation in order to reduce the misfit between simulated and measured observation values. The outcome of multi-realisation parameter adjustment process is an ensemble of parameter fields, all of which allow the model to adequately replicate observed system behaviour. These parameter fields can be considered samples of the posterior parameter probability distribution. By making a prediction using all members of this ensemble, the posterior probability distribution of that prediction is sampled.

The current tutorial demonstrates three possible IES workflows; these are also described in the Roadmap. The three workflows are distinguished by the manner in which initial parameter realisations are obtained. They are as follows:

1. Direct use of IES to calculate a posterior ensemble from a prior ensemble drawn from the prior parameter probability distribution (no calibration).
2. Calibrate first (using PEST/PEST++), and then sample a parameter probability distribution which employs the prior covariance matrix, but which is centred on the calibrated parameter field (post-calibration option 1).
3. Calibrate first (using PEST/PEST++), and then sample a parameter probability distribution which employs a linear approximation to the posterior covariance matrix, and which is centred on the calibrated parameter field (post-calibration option 2).

The computational cost increases in the same order as that in which the above workflows are listed; the rate of cost increase increases with the number of adjustable parameters. As you may recall from other GMDSI tutorials, the computational cost of conventional model calibration (attained through adjustment of a single parameter field using partial derivatives calculated using finite parameter differences) increases with the number of adjustable parameters. Unlike PESTPP-GLM and PEST, IES does not calculate derivatives using finite parameter differences. Instead, it calculates approximate partial derivatives from cross-covariances between parameter values and model outputs that are calculated using members of the ensemble. A major benefit that is forthcoming from this approach to calculating partial derivatives is that the number of model runs required to history-match an ensemble is independent of the number of adjustable parameters. This means that it is feasible for a model to employ a large number of adjustable parameters; conceptually, this reduces predictive bias at the same time as it guarantees that the uncertainties of decision-critical model predictions are not underestimated. At the same time, IES does not require that model outputs have the same level of precision as that required for finite-difference calculation of numerical derivatives. Hence numerical solver convergence settings can be loosened; this can reduce model run times.

However, as with most good things in life, the use of IES can sometimes incur costs. Due to the approximate nature of partial derivatives, it is not always possible to achieve as good a fit between model outputs and members of a calibration dataset as can be achieved using so-called "gradient methods" which employ partial derivatives. At the same time, the fewer realisations that IES employs, the more rank-deficient is the approximate sensitivity matrix that it calculates. The more observations that comprise a calibration dataset, the more this may compromise IES's ability to fit that dataset well. That being said, this may only be relevant in cases where temporal and spatial nuances of observation data are rich in prediction-pertinent information. IES can sometimes performs poorly where the history matching process is highly non-linear. In such cases, centring the initial parameter ensemble on calibrated parameter values (i.e. by employing one of the "calibrate first" workflows mentioned above) can often improve IES's performance. The "calibrate first" option has the added benefit of allowing a modeller to ascertain the minimum level of heterogeneity required to fit a calibration dataset, or whether that calibration dataset can be fit satisfactory at all. This allows him/her to thereby test the integrity of the model itself as a simulator of system behaviour.

Prior to the advent of IES, the "null space Monte Carlo" (NSMC) methodology was recommended for sampling the posterior parameter distribution. However, NSMC can be cumbersome to implement and is numerically more expensive. IES is therefore preferred. Nevertheless, the NSMC method (described in the Roadmap) is worth considering in circumstances where an excellent fit between model outputs and field measurements is required for all realisations comprising an ensemble. NSMC can be implemented either with PEST and associated utilities or with PESTPP-GLM. It will not be discussed further in the current tutorial.

2.1 Getting Ready

Regardless of the adopted workflow, some preparations must be made before using IES.

PEST/PEST++ Input Dataset

PEST input datasets can be read by members of the PEST++ suite, of which IES is a member. When reading a PEST control file, most programs of the PEST++ suite obtain the values of control variables that are unique to each such program from lines within the PEST control file that begin with the "++" string. PEST, and most of its associated utility support programs, ignore these lines. Interoperability of the PEST and PEST++ suites is thereby maintained.

For the current tutorial we will start off from the same PEST dataset prepared for the GMDSI calibration tutorial. Refer to this tutorial for all details of its configuration.

If you inspect the tutorial folder you will find two PEST control files. These are named *calibrated.pst* and *notcalibrated.pst*. (These are named very creatively, as you can see). These are copies of the PEST control files created during the GMDSI calibration tutorial. The only differences between these control files and those created in the calibration tutorial are that the initial parameter values assigned in the "parameter data" section of these respective PEST control files are different. As you may have guessed from the file names, *notcalibrated.pst* contains initial user-supplied parameter values, whilst *calibrated.pst* contains calibrated parameter values. Thus, ideally, these parameter values represent prior and posterior parameter means respectively.

In both of these PEST control files, observations are “weighted for visibility”; hence weights do not necessarily reflect measurement noise. Both of these PEST control files also feature Tikhonov regularisation.

Remove Regularisation

Because it performs uncertainty analysis rather than regularized inversion (i.e. calibration), prior information and accompanying covariance matrices should be removed from a PEST control file before it is used by IES. (As will be described shortly, these same covariance matrices can be used to support uncertainty analysis, rather than regularisation.) Regularization can be removed from a PEST control file using the SUBREG1 utility (as demonstrated in the GMDSI linear uncertainty tutorial).

1 Open a command line window in your working folder. Type the following and press <enter>:

```
subreg1 notcalibrated.pst ies-direct.pst
```

2 Inspect your working folder. You should see a new PEST control file named *ies-direct.pst*, from which regularisation has been removed. This control file will be used in the “direct application of IES” workflow.

3 Apply SUBREG1 to *calibrated.pst* as well. Name the resulting PEST control file *ies-postcalib1.pst*. (We will make a *ies-postcalib2.pst* PEST control file later on in the tutorial). Type the following and press <enter>:

```
subreg1 notcalibrated.pst ies-postcalib1.pst
```

4 Inspect your working folder. You should see a new PEST control file named *ies-postcalib.pst*, from which regularisation has been removed. This control file will form the basis of the two “calibrate first” workflows.

Ensemble Size

As previously mentioned, an “ensemble” refers to a set of parameter fields. Each of these parameter fields is referred to as a “realisation”. In general, the larger the ensemble, the better is the history-matching performance of IES. In practice, the number of realisations is limited by computing resources (more realisations require more model runs per iteration). The number of parameter fields comprising an ensemble should at least be larger than the dimensionality of the solution space of the inverse problem that is posed by the history-matching process. This ensures that the iterative ensemble smoother has access to the directions in parameter space that it requires in order to provide a good fit with the calibration dataset.

Normally, this number can only be guessed. It is usually less than the number of observations comprising a calibration dataset, and can never be greater than this. Furthermore, the dimensionality of the solution space decreases with the amount of noise that accompanies the dataset. If a Jacobian matrix is available (it may remain from a previous calibration exercise), the SUPCALC utility from the PEST suite can be used to assess solution space dimensionality. (Similar functionality is available from PyEMU.) In the GMDSI linear uncertainty tutorial a solution space dimensionality of 20 was calculated. Hence the ensemble should be comprised of 20 realisations at the very least. (Preferably at least a few extra realisations should be included.)

However, 20 samples of the posterior probability distribution of a prediction is insufficient to characterise this distribution. Hence the spread of predicted values will not reflect the true uncertainty of the prediction. For the current tutorial, because the model is fast and does not take long to run, we will use an ensemble size of 200. (Should you wish, you can always experiment with different ensemble sizes and compare the outcomes).

Initial Parameter Realisations

Initial parameter realisations can be generated internally by IES (the default). Alternatively, they can be supplied by the user in comma-delimited files, or in binary files. (“Post-Calibration Option 2” in the Roadmap describes this option).

In the current tutorial we will use IES to generate all prior realisations. To do this, we need a mechanism for characterization of prior parameter uncertainty. By default, IES assumes that parameter values in the “parameter data” section of the PEST control file comprise parameter mean values. Parameter variances/covariances can be specified in one of the following ways:

- By default, IES assumes that the difference between a (transformed) parameter’s upper and lower bounds is equal to 4 standard deviations of its prior probability distribution. This assumes that parameter bounds are span approximately the 95% confidence interval of the parameter value if it is normally distributed. If this option is taken, parameters are assumed to be statistically independent; hence it is assumed that there is no statistical correlation between them.
- Alternatively IES allows a user to directly specify parameter variances and covariances; a multiGaussian prior parameter probability distribution is assumed. These are passed to IES through either a parameter uncertainty file (described in the GMDSI linear uncertainty tutorial), or through an ASCII or binary covariance matrix file.

We will be using parameter uncertainty files in the current tutorial. If you inspect the tutorial folder you will see two parameter uncertainty files named *param.unc* and *post.unc*. You may recognise these files from the GMDSI linear uncertainty analysis tutorial. The former is a user-prepared file describing prior parameter uncertainties. The latter (together with the *post.mat* covariance matrix file) is generated by the PREDUNC7 utility; it characterises the post-calibration (posterior) parameter uncertainty under the assumption of model linearity. See the aforementioned tutorial for details on how to generate these files.

Observation Realisations

To generate realisations of observations, IES uses the weights associated with measurements listed in the “observation data” section of the PEST control file as a statistical descriptor of noise stochasticity. It assumes that the weight assigned to each observation is the inverse of the standard deviation of measurement noise associated with that observation.

Alternatively, the user can supply a comma delimited file (i.e. a CSV file) or an enhanced Jacobian matrix file (i.e. JCB file) containing observation realisations with noise already added according to his/her specifications. This is useful, for example, if the weighting scheme is designed to balance the initial contribution of all observation groups to the objective function and does not necessarily specify inverse standard deviations of measurement noise. These realisations can be generated using, for example, the RANDOBS utility from the PEST suite, or functions from the PyEMU suite. The current tutorial demonstrates the use of RANDOBS.

RANDOBS is described in Part 2 of the PEST Manual. It is very easy to use. RANDOBS generates random realisations of noise-enhanced observations for the use by the IES ensemble smoother. These realisations can be saved in a comma-delimited file (i.e. a CSV file) or in an enhanced Jacobian matrix file (i.e. a JCB file). RANDOBS reads an “observation weights file”. This can be any text file that has an “observation groups” section, such as a PEST control file for example. The “observation groups” section must start with the string “* observation groups”. Any line beginning with a “*” character terminates this section; alternatively, the end of the file terminates the section.

RANDOBS assumes that weights provided in the “observation groups” section of the file that it reads are equal to the inverse of the standard deviation of measurement noise. It uses this standard deviation to generate a random value of measurement noise for each observation centred on the value of that observation recorded in the observation weights file; a normal distribution is assumed. Note, however, that if the weight is zero for a particular observation, then the standard deviation of noise associated with that observation is assumed to be zero,; hence the value of the observation in all RANDOBS-generated realisations is the same, i.e. the value of the observation itself. Optionally, all observation weights can be multiplied by a factor before being used in the above manner for random realisation generation.

5 Make a copy of one of the PEST control files supplied in the tutorial folder. It doesn't matter which one. Name the copy *obsweights.dat*.

6 Open *obsweights.dat* in a text editor. Delete all sections except for the "observation data" section. You don't actually have to delete everything else, but it makes the file tidier and easier to navigate.

We can now proceed to replace observation weights (which reside in the third column of this file) with the inverse of the standard deviation of measurement noise associated with each observation. Let us assume the following:

- Head measurements have an uncertainty (i.e. a noise standard deviation) of 0.1 m;
- Stream flow measurements have an uncertainty of 10% of their measured values;
- Head differences have an uncertainty of 0.01 m.

7 Replace the observation weights in the third column of *obsweights.dat* with respective inverses of standard deviations (see in **bold** in the text box below) and then save the file. Your file should look similar to this:

```
* observation data
h1-3892_1 89.84313    1.00E+01    heads1
h1-1664_1 84.66009    1.00E+01    heads1
h1-162_1  85.48545    1.00E+01    heads1
h3-3521_1 94.85694    1.00E+01    heads3
h3-3988_1 94.84176    1.00E+01    heads3
h3-3417_1 91.62516    1.00E+01    heads3
h3-3829_1 90.0632     1.00E+01    heads3
h3-3357_1 88.31318    1.00E+01    heads3
h3-3594_1 89.08431    1.00E+01    heads3
h3-3892_1 89.7394     1.00E+01    heads3
h3-1152_1 84.79649    1.00E+01    heads3
h3-313_1  98.44412    1.00E+01    heads3
h3-209_1  90.80855    1.00E+01    heads3
h3-331_1  88.26986    1.00E+01    heads3
h3-707_1  86.52537    1.00E+01    heads3
h3-3956_1 87.48643    1.00E+01    heads3
h3-2270_1 84.60294    1.00E+01    heads3
h3-521_1  85.67934    1.00E+01    heads3
trib_1    -8385.727   1.19E-03    streams
total_1   -11233.4    8.90E-04    streams
dh3892_1  0.1037272   1.00E+02    headiff
p-trib_1  -6000       0.00E+00    pstreams
p-total_1 -6000       0.00E+00    pstreams
```

8 Open a command line window in your working folder, type the following and press <enter>:

```
randobs
```

9 You will be prompted for the name of the observation weights file. Reply as below and press <enter>:

```
Enter name of observation weights file: obsweights.dat
```

10 Next RANDOBS asks for a factor with which to multiply weights. We don’t wish to scale our weights, so simply reply with 1 and press <enter>:

```
Enter factor to apply to all weights: 1
```

11 Next you will be prompted for the number of realisations that RANDOBS must generate. This value should match the number of realisations of the parameter ensemble. (It can be greater than this if you like, but excess realisations are not read.) As previously mentioned, we will be using 200 realisations. Reply accordingly and press <enter>:

```
How many realisations to generate? 200
```

12 The next prompt asks if observation values in the observation weights file should be used as one of the realisations (without random noise enhancement). Reply as below and press <enter>:

```
Include initial observations as base realisation? [y/n]: n
```

13 At the next prompt, provide the name of the file which RANDOBS must write. The file name extension will determine the file format which RANDOBS employs to write this file. This can be “**.csv*” or “**.jcb*”. Let us use CSV file format; this has the benefit of being human-readable as well. Reply as below and press <enter>:

```
Enter name for output file: obs-ensemble.csv
```

14 You can choose whether RANDOBS records realisations in the CSV file per row or per column. In cases with many observations, recording per column can make it easier to open such a file in spreadsheet software such as EXCEL. For the tutorial it isn’t necessary, reply as below and press <enter>:

```
Realisations are rows or columns in csv file? [r/c]: r
```

15 Lastly, you will be prompted for the random number seed. By using the same random seed, the same random numbers will be generated on each occasion that RANDOBS is run based on the same input dataset. In our case this is not particularly relevant. Simply press <enter>:

```
Enter integer random number seed (<Enter> if default): <enter>
```

16 You should see a message stating that file *obs-ensemble.csv* was written ok. Inspect your working folder where you will see the new CSV file. Open it in any spreadsheet software (or text editor). Each observation realisation comprises a row of this file. Different noise-enhanced observations occupy different columns of this file.

Should you plot histograms of realisations for a particular observation, values will be distributed around its mean (the value provided in the observation weight file) with a standard deviation equal (or close, depending on the sample size) to the inverse of the weight that was assigned to the observation in the observation weights file.

2.1 Direct Use of IES

This workflow assumes that no calibration has been undertaken beforehand. We will be using IES to adjust samples of the prior parameter probability distribution until they become samples of the posterior parameter probability distribution. For the present example, this workflow is computationally frugal. However in the real world, the fit with historical data will often not be as good as that which we obtain in this workshop; as stated above IES can sometimes encounter difficulties if an inverse problem is highly nonlinear and/or if a model suffers numerical failure when supplied with random parameter sets.

Preparing the Control File

The current workflow is based on the PEST control file named *ies-direct.pst*. In principle, no further changes need be made to this control file in order to run IES. This is because members of the PEST++ suite supply default values for all non-user-specified control variables. Often default values are good-enough. (This does not excuse a user from reading the manual so that he/she is aware of the implications of using default values for control variables). As described previously, we wish to employ some non-default control variables in order to configure the size and manner in which prior parameter and observation ensembles are generated. We will also make a few other tweaks which tend to improve IES performance.

As has been previously mentioned, when reading a PEST control file, most programs of the PEST++ suite obtain the values of control variables that are unique to that program from lines within the PEST control file that begin with the "++" string. We shall now add several of these lines to *ies-direct.pst* in order to tweak our setup.

17 Open *ies-direct.pst* in a text editor and proceed to the bottom of the file. You should see the "model input\output" section with a list of template and instruction files. We will now add PEST++ control variables to the lines below this section.

18 As previously mentioned, we want an ensemble size of 200 realisations. Add the following line to the control file:

```
++ies_num_reals(200)
```

19 The "++" indicates that a PEST++ control variable follows; this is named *ies_num_reals*. The value of the variable is placed in brackets. In this case the integer *200*, corresponding to the number of realisations in the ensemble, is the value of the variable.

20 Next, we want to inform IES where it can read the ensemble of noise-enhanced observations which we generated using RANDOBS. To do so, add the following line to the control file:

```
++ies_observation_ensemble(obs-ensemble.csv)
```

21 Additionally, we want IES to sample the prior parameter distribution using the statistical specifications which we have provided in the parameter uncertainty file named *param.unc*. To do this, add the following line to the control file:

```
++parcov(param.unc)
```

Finally, let us configure two control variables which often improve IES's performance.

22 First, set the number of realisations used in testing and evaluation of different Marquardt lambdas using the keyword *ies_subset_size* to 7*.* In principle, a higher value for this variable may reduce the number of iterations that IES requires for solution convergence. However extra runs are required per iteration. Add the following line to the PEST control file:

```
++ies_subset_size(7)
```

Lastly, IES supports correlation-based, automatic adaptive localization. This functionality is activated by setting the *ies_autoadaloc()* control variable to *true*. Alternatively, a user can provide a localization matrix to define which parameters can be informed by which observations. The latter is preferred for cases in which the user knows *a priori* that certain observations cannot be sensitive to changes in specific parameters. For example, an observation of past groundwater head cannot be sensitive to a parameter that represents recharge occurring at a later time. The PEST++ user guide (provided in the tutorial folder) provides a succinct description of localisation in the IES context. It is good practice to include some type of localisation. Manual specification is preferred, however automatic adaptive localisation can often be sufficient; as well as this, it is easy to implement.

23 To activate automatic adaptive localisation add the following line to the PEST control file:

```
++ies_autoadaloc(true)
```

That is all there is to it. We are now ready to run IES with *ies-direct.pst*. The last few lines of the control file should now look like this:

```
++ies_num_reals(200)
++ies_observation_ensemble(obs-ensemble.csv)
++parcov(param.unc)
++ies_subset_size(7)
++ies_autoadaloc(true)
```

24 Actually, one last thing! This is specific to the tutorial and does not necessarily translate to real-world applications. As we (the authors) have already run IES with this case, we know that none of the workflows described below show much parameter improvement beyond 7 iterations. To avoid wasting too much time, we will constrain the maximum number of iterations by setting NOPTMAX to 7 in the “control data” section (at the top of the file) as shown in **bold in** the text box bellow:

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

Running IES

Programs of the PEST++ suite have in common the fact that they run a model many times. These runs can be undertaken in serial or in parallel. By undertaking these model runs in parallel, the time required for completion of an inversion or optimization task can be reduced considerably. Most modern computers have multiple CPUs; access to parallel computing facilities is becoming more common. Undertaking model runs in parallel will be the most common manner of PEST++ suite deployment.

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

Similar to what was required for PEST_HP in the GMDSI calibration tutorial, each agent must operate in a separate folder. This folder must contain a copy of the PEST control file. Setup is easiest if each folder also contains copies of all template, instruction and model batch files. So, for the current tutorial, each “agent folder” must contain a copy of the *pest* folder and the *runmodel* folder.

Start the Agents

As was stated above, each agent must run in its own distinct folder. In principle, there is no reason why one of these agents cannot run in the same folder as the manager; however it must not run in the same folder as another agent. For the sake of the tutorial (and tidy file management) we will keep these folders separate. The following steps assume that you will be running the manager and agents on the same computer. See the PEST++ user guide for instructions on how to configure setups that run over networks.

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

26 In the *agent_dir* folder, create another folder and name it *agent1.*

27 Copy both the *runmodel* and the *pest* folders into the *agent1* folder

28 Open a command line in the ..*\agent_dir\agent1\pest* folder.

29 Start an IES agent and tell it to read the *ies-direct.pst* control file. Do this by typing the following command, and by then pressing <enter>:

```
pestpp-ies ies-direct.pst /h %computername%:4004
```

To speed up the process, you will need to distribute the workload across as many parallel agents as possible. Perhaps you will want to use the same number of agents (or less) as you have available CPU cores. In practice, however, this is not ideal. You must also consider the amount of memory that is installed on your machine, and the fact that execution of each model will be slowed by the fact that they are all writing to the same disk. (Consider using a RAM disk; however this will remove a chunk of memory that is otherwise available to all of your model instances.)) Most personal computers (i.e. desktops or laptops) these days have between 4 and 10 cores. Servers or HPCs may generally many more cores than this.

30 In your *agent_dir* folder, create as many "agent" folders as you wish to use. For example, if you have four cores, create four folders named *agent1, agent2, agent3* and *agent4*. Repeat steps 26 to 29 for each of those folders. As you can see, being able to do this programmatically will make your life a lot easier – especially when dealing with many agents. For example, it is possible to automate this process using a batch file or a simple Python script (PyEMU even has a function that does all the dirty work for you).

At this point you should have several command line windows open, each with an agent waiting for instructions from a PEST++ manager.

Start the Manager

31 Return to the original *pest* folder in the tutorial root directory and open another command line window. To activate the IES manager, type the following and then press <enter>:

```
pestpp-ies ies-direct.pst /h :4004
```

32 IES will commence functioning, writing some outputs to the screen as it does so. From here you can monitor its progress as well as any error messages.

Depending on how many agents you are using, IES shouldn't take too long to run. After approximately 2000 model runs over 7 iterations IES has already fit observation data relatively well for most realisations in the ensemble. (If you had set NOPTMAX higher, IES would undertake further iterations; however the improvement in fit would not be significant). Seven iterations took approximately 12 minutes on an i9 Intel CPU with 10 agents. You may recall from the GMDSI calibration tutorial that calibrating this same model took over 40 min using the same computing resources.

Just as occurs with PEST_HP, PEST++ replenishes several output files with information that is worth monitoring in order to keep an eye on how it is performing. As long as the software you use to read these files does nor prevent use of the is non-exclusive, you can access them whilst IES is working. See the PEST++ user guide for detailed descriptions of all of these file types. Of note are:

- *ies-direct.log* – records the times at which various processing steps begin and end, as well as any error messages. It is useful for debugging.
- *ies-direct.phi.meas.csv* – records objective functions for each iteration; these are calculated using noise enhanced observation realisations (i.e. from the values listed in *obs-ensemble.csv*).
- *ies-direct.phi.actual.csv* – records objective functions for each iteration; these are calculated from the single set of observations that are listed in the PEST control file.
- The series of files named *ies-direct.N.par.csv*, *ies-direct.N.obs.csv* and *ies-direct.N.rei.csv.* In which *"N"* is the iteration number (i.e. 0, 1, 2,…, 7); these record parameters, simulated observations and residuals for all realisations in each iteration. If *N=0*, these pertain to the prior parameter distribution.

Let us start by taking a look at the evolution of the objective function over the course of IES's progress. The CSV files *ies-direct.phi.meas.csv* and *ies-direct.phi.actual.csv* contain records of objective function for each iteration calculated from the weighted residuals between simulated observations and observations recorded in the observation ensemble and in the control file, respectively. The progression of these objective functions per number of model runs is shown in Figure 2. Note that the values for the "base" realisation are the same in both plots. This is because no noise is added to the base realisation. Thus in both cases residuals are in fact calculated from observations in the PEST control file.

Calculating the objective function based on the weights in the *ies-direct.pst* control file and assuming the residual is equal to noise (see *obsweights.dat* and steps 6 to 7) shows that noise accounts for an objective function value of 81.6. As you can see in Figure 2, the majority of realisations achieve an acceptable fit within approximately 1000 model runs (4 iterations). As you may recall from the calibration tutorial, that is less than half the number of model comprising a single iteration that are required to calculate a Jacobian matrix.

For the particular case of this tutorial, the observations in the control file contain no noise; they are the direct output from a model run that represents "reality". In a real-world case, these observations would include noise as well. That is why the progress of the "base" realisation is the same in both of the plots appearing in Figure 2. The point where the plot for the base realisations departs from that of the rest of the ensemble in Figure 2 (right) denotes the point at which any improved fit with realisations other than the base is gained from "fitting noise". This distinctive behaviour of the base realisation does not characterize most real-world cases.

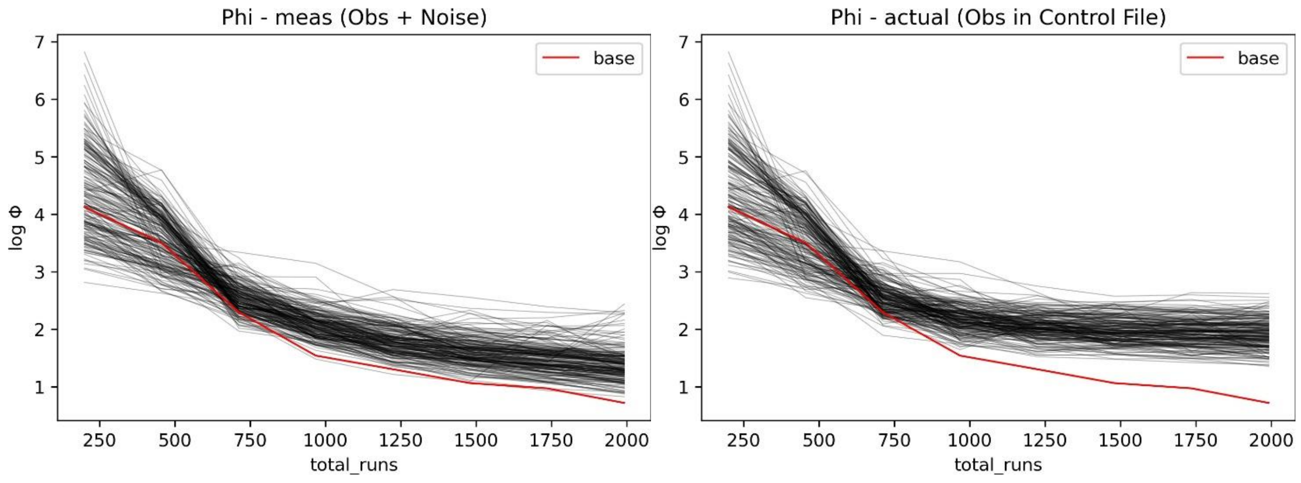

Figure 2 – Plot of objective function against number of model runs. The objective function is calculated from weighted residuals between simulated observations and observations recorded in the (left) observation ensemble and in the (right) control file. Recall that realisations of random noise are added to realisations comprising the observation ensemble (except for the base ensemble).

Figure 3 plots the progression of the statistical spread of the prediction of interest (observation named *p-trib_1* in the PEST control file, this pertaining to groundwater stream discharge) against IES iteration number. As you can see, by iteration number 4 or 5 the spread of model predictions has pretty much stabilized. We could have halted IES after 4 or 5 iterations and achieved pretty much the same outcome as we did after 7 iterations. As mentioned above, model-to-measurement fit at iteration 4 was already acceptable for the majority of the ensemble; hence it would have been reasonable to halt IES at that stage of its execution. (In fact we only let it carry on a bit longer for demonstration purposes).

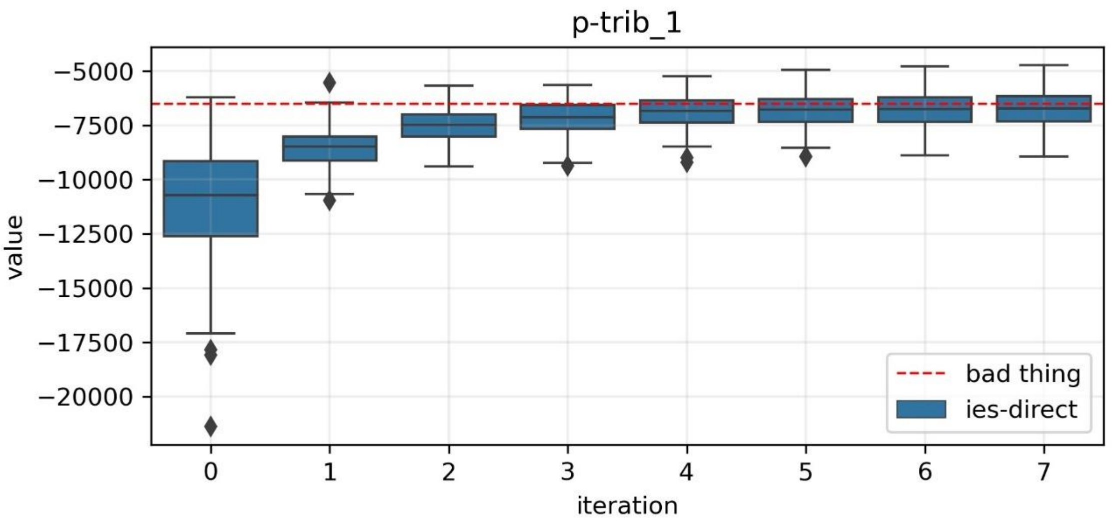

Figure 3 – Ensemble of model prediction *p-trib_1* per iteration; obtained with direct use of IES and no prior calibration.

Another (more common) way to display how the prior probability distribution of the prediction has been updated (i.e. conditioned) through history-matching is illustrated by Figure 4. In this figure, the grey bars depict the prior probability distribution of the prediction based on the prior parameter ensemble (i.e. parameters sampled in iteration 0). The blue bars denote the posterior probability distribution of the prediction based on the parameter ensemble achieved in iteration 7. (A similar plot for iteration 4 is pretty much identical). As expected, both the prior and the posterior probability distribution of the prediction include the “truth”. (“Truth” parameters were in fact sampled from the prior parameter probability distribution). In the present case the posterior predictive probability distribution is also centred on the “truth”. (This does not always happen, and in fact is a coincidence.) The narrower span of the posterior distribution shows that observations are useful in informing the prediction. (In real-world cases this is not always a foregone conclusion.) Unfortunately for our decision makers the posterior distribution of the prediction still includes outcomes which exceed the critical environmental flow threshold for frog safety; hence “bad things” may still happen. (See the red-coloured area in Figure 4). Therefore, the outcomes of this modelling exercise would not support implementation of the proposed groundwater abstraction (even though we, in our role as creators of this hypothetical reality, know that the “true” prediction does not result in a bad thing happening).

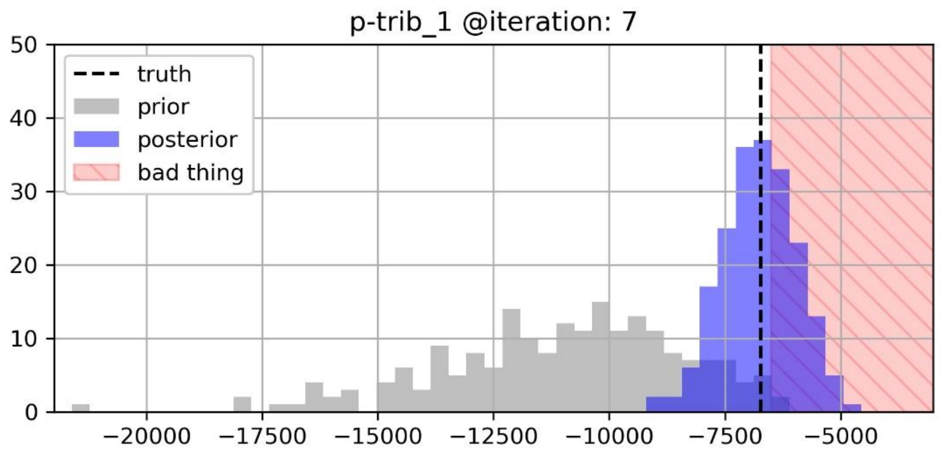

Figure 4 – Histograms of the prior and posterior probability distribution of model prediction of interest *p-trib_1*; obtained with direct use of IES and no prior calibration.

2.2 Calibrate First – Option 1

The following workflow assumes that calibration has been undertaken (using PEST, PEST_HP or PESTPP-GLM for example) and that parameters of minimum posterior error variance have thereby been obtained. However, sensitivities of model outputs to calibrated parameters (encapsulated in the Jacobian matrix) have not been re-calculated, as this would require that the model be run once for each adjustable parameter.

This workflow has an inherently higher computational cost than direct application of IES. However, it has the of providing a parameter set of “minimum error variance”. This parameter set encapsulates the heterogeneity “which must exist” to explain observed system behaviour embodied in the calibration dataset. This parameter set can provide insights into structures or heterogeneity which influence groundwater movement. Alternatively, it makes it very apparent to the user if parameters must adopt unrealistic values if model outputs are to fit observed system behaviour; this suggests that the structure of the model is informed by erroneous assumption. Furthermore, by centring the “prior parameter distribution” provided to IES on calibrated values, IES will probably converge faster, particularly in difficult cases where the relationship between model outputs and parameters is highly nonlinear.

Recall that we previously prepared a PEST control file named *ies-postcalib1.pst*. This file contains parameter values of minimum error variance as initial parameters. These are listed in the “parameter data” section of this file.

33 Open *ies-postcalib1.pst* in a text editor.

34 Assign a value of 7 to NOPTMAX.

35 Add exactly the same PEST++ control variables that were assigned to *ies-direct.pst* in steps 18 to 0 and save the file.

36 We are now ready to run IES with *ies-postcalib1.pst*. Prepare the agents and manager in the same manner as before (see steps 26 to 31). The command to run each agent in its respective folder becomes:

```
pestpp-ies ies-postcalib1.pst /h %computername%:4004
```

37 The command to run the manager becomes:

```
pestpp-ies ies-postcalib1.pst /h :4004
```

Once again, IES will run for 7 iterations. If allowed to, it would continue for about 20 iterations. However objective functions do not improve significantly after the seventh iteration. In fact, as you can see from Figure 5, the majority of realisations comprising the ensemble have achieved an acceptable fit with the history-matching dataset after approximately 1000 model runs (4 iterations). The "base" realisation has a markedly lower objective function as (1) it is based on the calibrated parameters and (2) residuals are calculated from observations in the PEST control file which are not supplemented with random realisations of measurement noise. (The objective function associated with this realisation decreases because the previous calibration exercise was halted when a user-defined target measurement objective function was achieved. This limit is not imposed by IES.)

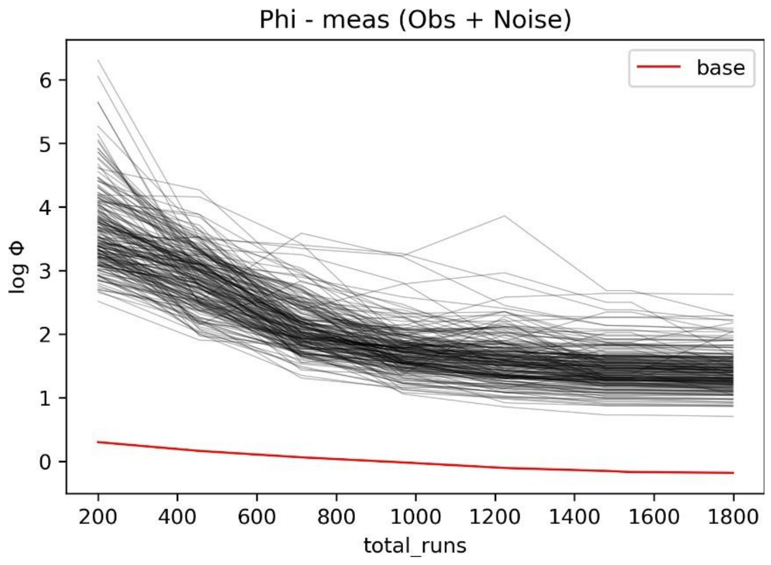

Figure 5 – Plot of objective function against number of model runs. The objective function is calculated from weighted residuals between simulated observations and observations recorded in realisations comprising the observation ensemble. Sampling of the posterior parameter distribution was undertaken by calibrating first and then commencing the IES process from random parameter sets centred on calibrated parameter field.

Figure 6 shows the prior and posterior histograms for prediction *p-trib_1* after 7 iterations. The posterior histogram is similar to that obtained using IES directly. T The prior histogram is different, however. It is centred on calibrated parameter values in Figure 7. The spread of the prediction's posterior is similar to that obtained with IES directly.

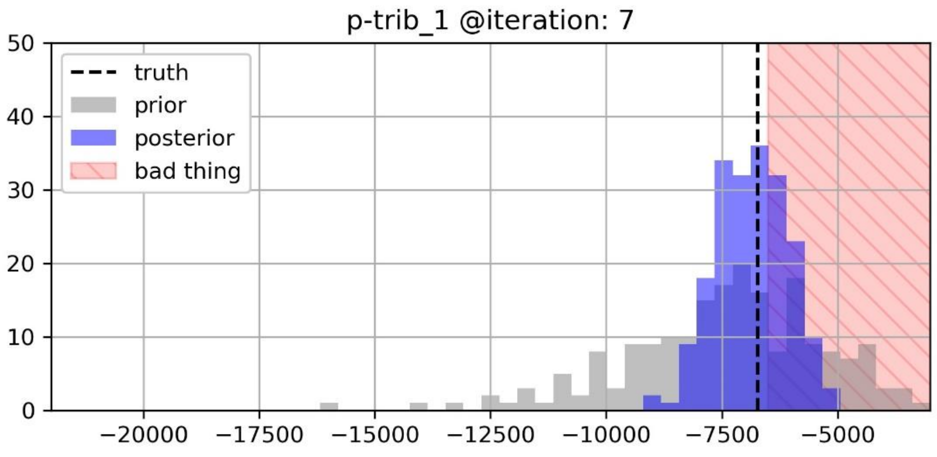

Figure 6 – Prior and posterior histograms of model prediction *p-trib_1*. These were obtained by calibrating first and then commencing the IES process using a covariance matrix that reflects prior parameter spread, but is centred on the minimum error variance parameter field obtained through model calibration.

2.3 Calibrate First – Option 2

This workflow assumes that model calibration has been undertaken, and that parameters of minimum error variance have thereby been obtained. After calibration, sensitivities were recalculated; finite differences were taken with respect to the calibrated parameter set. Lastly, linear analysis was undertaken to calculate a posterior parameter covariance matrix using the PREDUNC7 utility. This matrix was recorded in a file named *post.mat*. The parameter uncertainty file named *post.unc* links this matrix file to parameters listed in the PEST control file.

This workflow has the highest computational cost. However it has the benefits described above of attaining a parameter field of minimized error variance, and hence (presumably) a parameter field that embodies the minimum amount of heterogeneity required to fit the calibration dataset. Also, by providing IES with parameter realisations that are samples of a linear approximation to the posterior parameter probability distribution, its task in sampling this distribution itself is made much easier.

A single change is required to *ies-postcalib1.pst* in order to implement this workflow.

38 Make a copy of *ies-postcalib1.pst*. Name the copy *ies-postcalib2.pst*.

39 In *ies-postcalib2.pst*, replace the value assigned to the *parcov()* control variable from *param.unc* to *post.unc*; as shown below:

```
++parcov(post.unc)
```

40 The PEST++ variables in *ies-postcalib2.pst* should now look like this:

```
++ies_num_reals(50)
++ies_observation_ensemble(obs-ensemble.csv)
++parcov(post.unc)
++ies_subset_size(7)
++ies_autoadaloc(true)
```

We are now ready to run IES with *ies-postcalib2.pst*.

41 Prepare the agents and manager in the same manner as before (see steps 26 to 31). The command to run each agent in its respective folder becomes:

```
pestpp-ies ies-postcalib2.pst /h %computername%:4004
```

42 The command to run the manager becomes:

```
pestpp-ies ies-postcalib2.pst /h :4004
```

Once again, IES will run for 7 iterations. If allowed to, it would continue for a few more iterations. However the ensemble objective function does not improve significantly after the third iteration (969 model runs). A comparison of Figure 7 with Figure 2 and Figure 5 shows that IES converges faster when run in this manner. In the present case, convergence is hastened by only a moderate amount. In highly nonlinear cases, differences in convergence speed can be considerable.

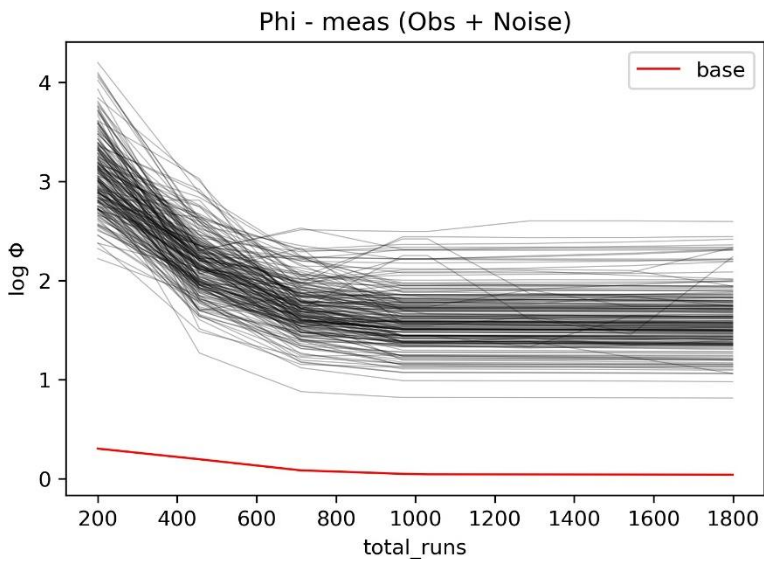

Figure 7 – Plot of objective function against number of model runs. The objective function is calculated from weighted residuals between simulated observations and observations recorded in realisations comprising the observation ensemble. IES sampling of the posterior parameter distribution was achieved by calibrating first and then sampling from a linear approximation to the posterior parameter probability distribution.

The posterior probability distribution of the prediction (Figure 8) is similar to that obtained through the previous two workflows demonstrated in this tutorial. In the present case, the prior probability distribution of the prediction has a smaller spread than those which were previously employed; it similar to the posterior distribution achieved by IES.

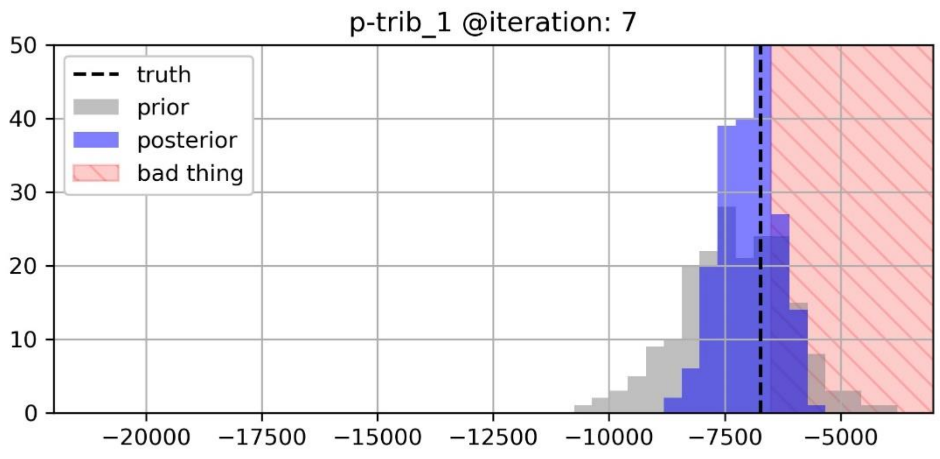

Figure 8 – Histograms of the prior and posterior probability distribution of model prediction of interest *p-trib_1*; obtained by calibrating first and starting from the posterior parameter probability distribution. Prior and posterior histograms of model prediction *p-trib_1*. These were obtained by calibrating first and then commencing the IES process using a linear approximation to the posterior parameter distribution.

Progression of the uncertainty interval of the model prediction is plotted against iteration number for the three workflows demonstrated herein in Figure 9. As you can see, all three workflows yield similar outcomes in terms of the posterior probability distribution of the prediction. The “calibrate first” workflows result in faster ensemble convergence, particularly if prior realisations are sampled from a linear approximation to the posterior parameter probability distribution. However, this workflow is accompanied by a higher computational cost, particularly where many parameters are involved.

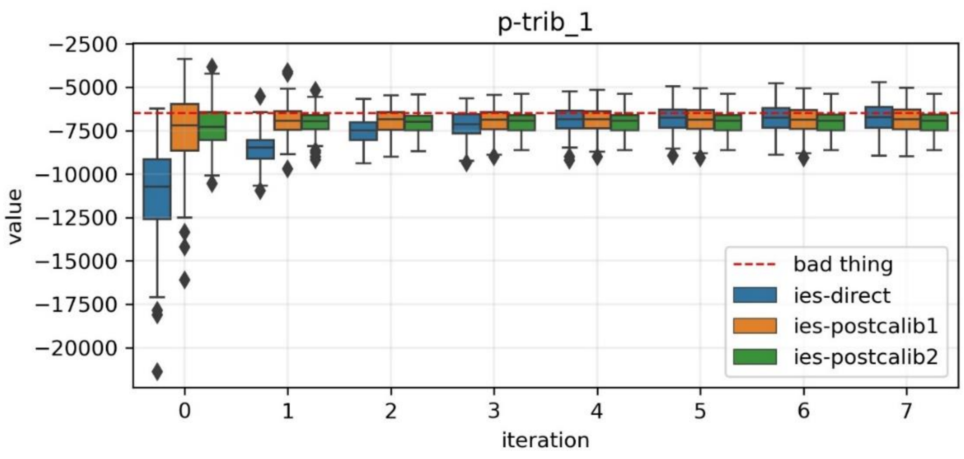

Figure 9 – Ensembles of model prediction *p-trib_1* plotted against iteration number for the three IES workflows demonstrated in the current tutorial.

2.1 Final Words

Non-linear uncertainty analysis undertaken for this tutorial demonstrates the use of the iterative ensemble smoother implemented in the PESTPP-IES program. The current tutorial provides only brief introduction to IES. Hopefully it will also provide you with an incentive to continue exploring opportunities offered by IES for solving real-world decision-support modelling problems with high levels of numerical efficiency.

This tutorial has demonstrated that direct application of IES can achieve acceptable levels of model-to-measurement fit with a fraction of the computational burden associated with conventional calibration methods. In general, provided a context-specific minimum number of realisations comprise the parameter ensemble, this computational burden does not increase with parameter count. This is not the case for conventional calibration where a Jacobian matrix must be filled using finite parameter differences..

Nevertheless, “calibrate first” workflows can be useful in cases where IES struggles to history-match observation data. This can occur if the relationships between model outputs and parameters is highly non-linear, or if a model is numerically temperamental when supplied with random parameter fields. Conventional calibration provides the added benefit of calculating a full-rank Jacobian matrix; this can serve as a basis for linear analysis. As well as yielding estimates of parameter and predictive uncertainty, linear analysis can also yield value-added quantities such as the worth of existing or yet-to-be-acquired data, and contributions to predictive uncertainty made by different parameters and/or groups of parameters. Value-added quantities such as these are difficult to calculate when the non-linear relationships between model outputs and parameters are acknowledged and accommodated through the use of ensembles.

Some of the GMDSI Worked Examples demonstrate applications of IES in a real-world context. As has been mentioned, the PEST Roadmap 13: Non-Linear Analysis provides an overview of the underpinning theory, along with some practical advice. Videos available from the GMDSI Webinar series also discuss non-linear uncertainty analysis and provide insight into their application in real world examples.

Groundwater Modelling
Decision Support Initiative

gmdsi.org

NATIONAL CENTRE FOR GROUNDWATER RESEARCH AND TRAINING