# Four Ways to Explore Model Predictive Uncertainty

A GMDSI Workshop

November, 2022

# 1. Introduction

This brief tutorial demonstrates four ways to explore the posterior (i.e. history-match-constrained) uncertainties of two predictions made by a simple, highly-parameterised, model. These are:

1. linear analysis;
2. pseudo-nonlinear analysis;
3. iterative ensemble smoother;
4. data space inversion.

Most of the steps that are demonstrated in this tutorial use programs of the PEST suite. However the iterative ensemble smoother (PESTPP-IES) belongs to the PEST++ suite.

Note that PyEMU and PESTPP-GLM (from the PEST++ suite) provide alternative means to accomplish linear uncertainty analysis.

PEST and associated utilities can be downloaded from <https://pesthomepage.org/programs>

The PEST++ suite can be downloaded from <https://github.com/usgs/pestpp>

PyEMU can be downloaded from <https://github.com/pypest/pyemu>

This tutorial does not provide instructions on the use of MODFLOW, or of PEST utility programs that support the use of MODFLOW. Its purpose is to demonstrate calibration and uncertainty analysis. The concepts and procedures that are demonstrated herein can be applied to any model, whether or not it is a groundwater model.

# 2. The Model

The model that is used in the present example simulates steady-state flow of groundwater through a single-layered medium with heterogeneous hydraulic conductivity. The log (to base 10) mean of hydraulic conductivity is 0; this corresponds to an actual hydraulic conductivity value of 1.0 m/day. The variance of the log (to base 10) of hydraulic conductivity is 0.2; its prior standard deviation is therefore 0.447. Porosity is uniformly 0.1. The model will be calibrated using heads measured in 12 observation wells.

We will explore the uncertainties of two predictions. One of these is the location at which a particle that is released near the top of the model domain exits the model through its bottom boundary. This is a fixed head boundary. The other is the time that it takes for the particle to reach this boundary. The situation is depicted in Figures 2.1 and 2.2.



Figure 2.1. Model grid, hydraulic conductivity, observation wells and path of a particle.

The model that PEST runs is encapsulated in a batch file named *model.bat*. The groundwater model is MODFLOW6. Particle movement is simulated using MODPATH7. The MODFLOW6 grid is structured; all cells are square with sides of 10 m.

Inspect *model.bat*. This has features in common with most models that PEST must run. The simulator is preceded by a parameter pre-processor (in this case PLPROC), and followed by an observation post-processor (in this case MOD2OBS\_DBL). Many PEST-support utility programs are designed to play these pre/postprocessing roles. PLPROC is used for pilot points parameterization; pilot points that are used for parameterisation of the present model are shown in Figure 4.1. Meanwhile, MOD2OBS\_DBL is used for processing of MODFLOW-calculated heads arrays. The “DBL” version of MOD2OBS is used instead of the normal version of this program because MODFLOW6 records its system state output files in double precision. MOD2OBS (and MOD2OBS\_DBL) read a MODFLOW-generated binary heads file, in this case *rect.hds*. They undertake temporal and spatial interpolation of MODFLOW-generated heads to the times and locations of field measurements. In the present example, field measurements are recorded in a file named *observ.smp*. The format used for this file is common to many PEST utilities; it is a “site sample file”.



Figure 2.2. Groundwater head contours and particle travel path superimposed on heterogeneous hydraulic conductivity (i.e. K) field.

# 3. Adding Regularisation to a PEST Control File

## 3.1 Some Background

We will start with an already-built PEST control file; see *case.pst*. There are many ways to build a PEST control file. Its specifications are explained in the PEST manual; so you can build it with a text editor. However there are easier ways to build it. These include:

* a graphical user interface that supports PEST, for example Groundwater Vistas, Visual MODFLOW, GMS, or ModelMUSE;
* PyEMU; and
* utilities such as PESTPREP1 and PESTPREP2 that are supplied with the PEST Groundwater Utility Suite.

Regardless of how it is built, it is good to know your way around a PEST control file so that you can edit parts of it in order to tailor the history-matching process to suit your needs.

Initial parameter values are provided in the “parameter data” section of *case.pst*. Ideally, initial parameter values should be expected parameter values as defined by the prior parameter probability distribution. In this case they are all 1.0 m/day. This is not to say that we expect the parameter field to be uniform. What we are saying is that a value of 1.0 m/day lies at the centre of the prior parameter probability distribution at all places within the model domain. These values therefore provide “fall-back positions” from which Tikhonov regularisation will seek minimum departure.

In the “observation data” section of *case.pst*, some observations have weights of zero. These are, in fact, “predictions”. On this occasion, predictions are made using the same model as that on which history-matching is based. On other occasions, a special PEST control file should be devoted to running the model for predictive purposes.

Note also that singular value decomposition is employed to solve matrix equations that are formulated by PEST to implement history-matching. This is recommended practice. See the “singular value decomposition” section of *case.pst* wherein the EIGTHRESH variable is set to 5×10‑7. This low setting for EIGTHRESH prevents the occurrence of numerical problems that can attend attempts to invert near-singular matrices. Meanwhile, as we shall demonstrate shortly, regularisation is implemented using a Tikhonov scheme.

The RLAMFAC variable is set to -3. (RLAMFAC is the second variable on the sixth line of *case.pst*.) This is also recommended practice. However if you use PEST\_HP to calibrate a model, it can evaluate the best Marquardt lambda settings itself; it ignores variables such as RLAMFAC which govern selection of trial Marquardt lambdas.

As stated above, parameterization is based on pilot points. The pilot points file is *hk.pts*. The template of this file is *hk.tpl*.

## 3.2 Getting Ready for Regularised Inversion (i.e. Calibration)

Regularisation specifications and control variables can be added to a PEST control file using the ADDREG1 utility. ADDREG1 provides a prior information equation for each parameter featured in the PEST control file. This equation informs PEST that the preferred value of each parameter is its initial value. This will therefore be its estimated value unless information that is contained within the calibration dataset dictates otherwise.

Run ADDREG1 as follows.

|  |
| --- |
| **> addreg1 case case1** |

Regularisation of pilot point parameters should generally include a covariance matrix. The purpose of this covariance matrix is to ensure that departures from preferred parameter values arise in ways that are “legitimate” from the point of view of the prior parameter probability distribution. Ideally, this covariance matrix should therefore specify prior parameter uncertainties, including prior parameter spatial correlation.

If a modeller is happy with the assumption of “stationarity” (a geostatistical concept which states that spatial parameter correlation is independent of location), a covariance matrix can easily be built from a variogram using the PPCOV utility from the PEST Groundwater Utility Suite. PPCOV obtains the geostatistical structure which describes this variogram from a “structure file”. In the present case, this file is named *struct.dat*. The variogram which is provided in *struct.dat* is that from which the reality parameter field depicted in Figures 2.1 and 2.2 was sampled. Because our case is synthetic, we know the “variogram of reality”. In real-world cases we must make an educated guess.

An inspection of *struct.dat* reveals that the “true” variogram (referenced through the *structure1* structure that is specified in this file) pertains to the log of hydraulic conductivity parameters rather than to their native values. This variogram is of the exponential type, has a sill of 0.2 and an “*a*” value (about a third of its range) of 200m. (Note that regularised inversion – that is, model calibration – will not suffer too much if the prior covariance matrix which is used by PEST is “wrong”. Its main job is to ensure that the calibrated parameter field is smooth, and does not exhibit artificial “bumps, lumps and bullseyes” that are centred on pilot points.)

Run PPCOV as follows.

|  |
| --- |
| **> ppcov**  Program PP2COV prepares a covariance matrix file for pilot point parameters  based on a geostatistical structure file.  Enter name of pilot points file: ***hk.pts***  - data for 104 pilot points read from pilot points file hk.pts  Enter minimum allowable separation for points in same zone: ***0***  Enter name of structure file: ***struct.dat***  Enter structure to use for pilot point zone 1: ***structure1***  Enter name for output matrix file: ***cov.mat***  Enter pilot point prefix for parameter name (<Enter> if none): ***k\_***  Filling covariance matrix....  - file cov.mat written ok. |

PEST provides a number of mechanisms for linking elements of a covariance matrix to parameters. One such mechanism is through parameter and pilot point names. Even though PPCOV asks for the linkage between pilot point and parameter names (to which the answer “k\_” is provided above), this mechanism is not used in the present case for linking parameter-specific prior information equations to elements of the PPCOV-generated covariance matrix. Instead, linkage is by order. So it is important to ensure that the ordering of pilot points in the pilot points file (*hk.pts* in the present case) is the same as the ordering of prior information equations that reference respective pilot point parameters in the PEST control file. Because prior information equations were added to the PEST control file by ADDREG1, this occurs automatically if the ordering of pilot points in *hk.pts* is the same as the ordering of respective parameters in the “parameter data” section of the PEST control file. This is easy to ensure.

The PEST Groundwater Utility Suite includes other PPCOV-type utilities. These are PPCOV3D, PPCOV\_SVA and PPCOV3D\_SVA. The last two of these programs can build a covariance matrix based on spatially varying variograms. This can accommodate the presence of different geological units, local geological structure, and meandering alluvia within a model domain.

Now edit *case1.pst*. Do the following.

1. Insert the name “*cov.mat*” in the *observation groups* section of *case1.pst* opposite the observation group name “*regul\_k*”.
2. Set the target measurement objective function (i.e. the value of the PHIMLIM control variable) to 12.0, and the value of the variable next to it (i.e. PHIMACCEPT) to 12.1. These are the first two variables on the first line of the *regularisation* section of *case1.pst*.

The *observation groups* section of *case1.pst* should now look like this.

|  |
| --- |
| \* observation groups  heads  time  distance  regul\_k cov.mat |

The *regularisation* section of *case1.pst* should now look like this.

|  |
| --- |
| \* regularisation  12.0 12.1 0.1000000  1.0 1.0e-10 1.0e10  1.3 1.0e-2 1 |

Note that all nonzero weights in the *observation data* section of *case1.pst* are set to 10. These were inherited from *case.pst*. This is (conveniently) the inverse of the standard deviation of noise that is associated with measurements of groundwater head. Because the standard deviation of measurement noise is 0.1 m and the weight that is assigned to each observation is 10.0, the average value of each head residual achieved through history-matching of modelled to measured heads should be 1.0. The total objective function should therefore be approximately equal to the number of head observations, namely 12.0. This is the reason why we set PHIMLIM to 12.0.

In the real-world, a modeller does not know the standard deviation of “measurement noise”; he/she finds this out when calibrating the model. Under these circumstances, set PHIMLIM to a very low value, for example 1.0E-10; set PHIMACCEPT to 1.1E-10. Then calibrate the model to see the level of model-to-measurement fit that PEST can attain. Meanwhile, set the third variable on the first line of the *regularisation* section of the PEST control file (its name is PHIMFRAC) to 0.1. This ensures that the target measurement objective function is always set internally to 0.1 of the current value of the measurement objective function. Regularisation is therefore always active; this enables reasonableness of parameter values. If you like, you can then run PEST again with the target measurement objective function set 5% to 10% higher than the best that PEST was able to achieve. This normally ensures a smooth parameter field. Alternatively, undertake a single PEST run and adopt a parameter set from a previous iteration as the “calibrated” parameter set. Meanwhile, turn on the PARSAVEITN variable in the *control data* section of the PEST control file in order to ensure that PEST saves an iteration-specific parameter value file at the end of every iteration of the inversion process.

# 4. Model Calibration

## 4.1 Checking

Check the entire PEST input dataset using the PESTCHEK utility.

|  |
| --- |
| **> pestchek case1** |

Ignore any PESTCHEK warnings.

PESTCHEK checks the integrity of all aspects of a PEST input dataset. In doing so, it ensures that all template and instruction files that are cited in a PEST control file are present, and that parameter and observation names are consistent between them and the PEST control file.

## 4.2 Calibrating with PEST

Run PEST using the command:

|  |
| --- |
| **> pest case1** |

(If you are in a hurry, there is no need to wait until PEST completes the calibration process. You can stop it as soon as the measurement objective function approaches its target value of 12.0.)

The calibrated hydraulic conductivity (i.e. K) field and particle path are shown in Figure 4.1. The calibrated K field allows the model to match head observations very well. However it is considerably less heterogeneous than the real K field.



Figure 4.1. Calibrated K field and particle path. Pilot points are shown as diamonds.

Predictions made by the calibrated model are shown in Table 4.1, where they are compared with the true values of these predictions. Predictions made by the calibrated model can be read from the run record file *case1.rec* or from the residuals file *case1.res*. (Because predictions are made by the same model that is used for calibration, inspection of their values is easy.)

|  |  |  |
| --- | --- | --- |
| **Prediction** | **Reality K field** | **Calibrated K field** |
| Particle travel time (days) | 3256.2 | 6840.8 |
| Particle emergence easting (m) | 206.8 | 225.9 |

Table 4.1. Predictions made using the reality K field and the calibrated K field.

Notice how wrong is the particle travel time prediction made by the calibrated model. This does not mean that the model is “bad”, for there is no mathematical reason why a “calibrated model” should make a correct prediction. A model is only “bad” when the correct prediction does not lie within the uncertainty limits that are quantified using that model. We will quantify predictive uncertainty shortly.

## 4.3 Calibrating with PEST\_HP

We will now calibrate the model again using PEST\_HP. This provides an opportunity to illustrate some useful PEST\_HP settings. There is really no need to use PEST\_HP instead of PEST for this simple case. However, where model run times are long, where the calibration process is highly nonlinear, where a model exhibits some numerical problems, and/or if you have access to many computing cores, PEST\_HP may do a much better job than PEST.

Make a copy of *case1.pst*.

|  |
| --- |
| **> copy case1.pst case1a.pst** |

Now edit the “control data” section of *case1a.pst* in accordance with the bold script in the figure below.

|  |
| --- |
| \* control data  restart regularisation  104 14 1 104 4  1 2 single point 1 0 0  10.0 -3.0 0.3 0.03 10 ***999 uptestmin=20***  10.0 10.0 0.001  0.1 noaui  30 0.005 4 4 0.005 4  1 1 1 |

The “999” string (this is the JACUPDATE variable) informs PEST\_HP that it must do Broyden Jacobian updating. Hence two cycles of lambda testing are performed during each iteration of the inversion process – one with a Jacobian matrix calculated using finite parameter differences, and one with a Jacobian matrix that has been improved through lambda testing. Sometimes, in highly nonlinear settings, this can yield dramatic improvements in inversion performance.

The string “uptestmin=20” informs PEST\_HP that it must devote at least 20 model runs to lambda testing during each of these lambda testing cycles. This ensures that lambda testing is thorough. PEST\_HP will do more lambda tests than this if there are more computing cores at its disposal. The number of lambda tests can be limited (often a good idea) using the UPTESTLIM variable.

(PEST\_HP undertakes lambda testing using a different algorithm from that used by PEST; its algorithm works best where many model runs are being conducted in parallel.)

Use PESTCHEK to ensure that you made the above changes correctly. Ignore the warnings.

|  |
| --- |
| **> pestchek case1a** |

Open up another command-line window that is focussed on the current working folder. One way to do this is to type the following command at the screen prompt.

|  |
| --- |
| **> start cmd** |

Run the PEST\_HP agent in this new command-line window. Type the following command.

|  |
| --- |
| **> agent\_hp case1a /h %computername%:4004** |

*computername* is an environment variable whose value is the hostname of your computer. The enclosing “%” symbols inform the command-line interpreter that the string between them is an environment variable.

Now, in the original command-line window, start PEST\_HP using the following command. (Be sure to leave a space between “/h” and “:4004”.

|  |
| --- |
| **> pest\_hp case1a /h :4004** |

You can stop execution of PEST\_HP at any time using <Ctl-C>.

# 5. Getting the Weights Right

## 5.1 Background

For the analyses that follow, we need a PEST control file in which weights are the inverse of the standard deviation of measurement noise. For reasons which have already been stated, when weights are “correct” in this way, the calibration objective function should be roughly equal to the number of non-zero-weighted observations which comprise the calibration dataset.

On this occasion we were able to equip PEST with correct weights – and a correct target measurement objective function – because we know the standard deviation of measurement noise. In most history-matching cases we do not know the level of measurement noise. This is because “measurement noise” is a loose term for “whatever is responsible for our inability to fit field data perfectly” using a model. In many real-world cases it is the imperfect and approximate nature of numerical simulation that contributes most to model-to-numerical misfit. When we calibrate a model, we find out how much “noise” there is through the fit that we achieve with the calibration dataset, or through the fit that we will tolerate because acquisition of a better fit will result in parameter fields which we do not like.

Let us pretend that, in the present example, we do not know the “correct” weights, and that we have to back-calculate them from the objective function that we achieved through history-matching. Of course, in the present case we will obtain the same weights that we started with. In most real-world cases we will not.

As well as ensuring that weights are correct, we need to modify the PEST control file in some other ways. We will remove regularisation from it. We will also modify its initial parameter values so that these are calibrated parameter values. After calculating a Jacobian matrix based on these calibrated parameter values we will be ready for linear predictive uncertainty analysis.

## 5.2 Preparing a New PEST Control File

We will use the calibrated parameter values that we obtained using PEST rather than those which we obtained using PEST\_HP because the former are slightly better. (Broyden Jacobian updating can accelerate an inversion process, and can maintain progress of that process in difficult numerical circumstances; however it can sometimes introduce some spurious heterogeneity into a calibrated parameter field.)

PEST-calibrated parameter values are stored in file *case1.par*; this file was written by PEST when it calibrated the model. We will insert these as initial values into a new PEST control file named *case2r.pst*. The command is:

|  |
| --- |
| **> parrep case1.par case1.pst case2r.pst** |

Regularisation can be removed from this file using the command:

|  |
| --- |
| **> subreg1 case2r case2** |

Now set the NOPTMAX variable to 0 in *case2.pst*. This informs PEST that it must run the model once and then cease execution. NOPTMAX is the first variable on the 9th line of a PEST control file. Alter it from 30 to 0, and then run PEST.

|  |
| --- |
| **> pest case2** |

PEST runs the model once using optimised parameters. It then calculates the objective function before ceasing execution. It records the objective function, along with other information, in the run record file *case2.rec*. It also writes the objective function to the screen. It is 12.0, as it should be. This is because weights are already “correct” in file *case2.pst*. As stated above, in real-world PEST usage, this will not happen often.

So let us pretend that the weights are not correct, and that we would like to alter them so that they are correct. Many PEST utilities, and the PESTPP-IES ensemble smoother, require that weights listed in a PEST control file be “correct”. As stated above, they are “correct” when the objective function is roughly equal to the number of observations. The PWTADJ2 utility adjusts weights in order to achieve this – for the objective function as a whole, and/or for the component of the objective function that pertains to each observation group. Just so you know how to use PWTADJ2, its use will be demonstrated. Run the following command.

|  |
| --- |
| **> pwtadj2 case2 case3 g** |

PWTADJ2 reads the objective function that is listed in the run record file *case2.rec.* On the basis of this objective function, it calculates adjustment factors for weights that are provided in *case2.pst*. PWTADJ2 then writes a new PEST control file named *case3.pst* in which these adjustment factors are applied. In this special case, the weights in *case3.pst* are unchanged from those in *case2.pst*. In fact the only thing that is different about *case3.pst* is that the NOPTMAX variable is set to 50 in *case3.pst*. PWTADJ2 does this in case you forget to do it yourself.

Now you know how to use PWTADJ2.

## 5.3 Obtaining a Jacobian Matrix

Set NOPTMAX to -2 in the “control data” section of *case3.pst*. This section should then appear as follows. (The NOPTMAX variable is highlighted.) This NOPTMAX setting tells PEST that it should run the model only enough times to calculate a Jacobian matrix; it should then cease execution.

|  |
| --- |
| \* control data  restart estimation  104 14 1 0 3  1 2 single point 1 0 0  10.0 -3.0 0.3 0.03 10  10.0 10.0 0.001  0.1 noaui  ***-2*** 0.005 4 4 0.005 4  1 1 1 |

Now run PEST using the following command.

|  |
| --- |
| **> pest case3** |

PEST records the Jacobian matrix in a file named *case3.jco*. This is a binary file. PEST-support utilities such as JACWRIT and JCO2MAT allow you to translate this into a text file so that you can inspect its contents.

We now have:

* a PEST control file in which weights are correct, and
* a corresponding Jacobian matrix file.

So we are ready to undertake linear analysis. Note that the integrity of linear analysis does not rely on the fact that the PEST control file that was used to calculate the Jacobian matrix contains calibrated parameter values. The outcomes of linear uncertainty analysis are (in theory at least) independent of model parameter values. However because the operation of most models is not linear with respect to its parameters, it is best to calculate the Jacobian matrix using calibrated parameter values if this is possible.

It is also important to note that while linear analysis provides parameter and predictive uncertainty intervals, it does not provide numbers that are at the centres of these intervals. These are provided by model calibration.

# 6. Linear Uncertainty Analysis

## 6.1 Prior Parameter Uncertainties

In Section 3 of this tutorial we built a covariance matrix for use in regularisation. This matrix describes prior parameter uncertainties. So we can also use this matrix for uncertainty analysis.

Programs and utilities from the PEST and PEST++ suites require that prior parameter uncertainties be provided in a “parameter uncertainty file”. The specifications for this type of file are described in Part 2 of the PEST manual. This file can cite an unlimited number of covariance matrices; these may pertain to parameters in different layers of a model, or within different zones. A parameter uncertainty file can also provide lists of prior parameter standard deviations; this is appropriate for parameters which exhibit no correlation (spatial or otherwise) with eachother.

In the present case, our attention is focussed on a single set of parameters whose prior uncertainties are described by a single covariance matrix. These parameters are K values attributed to pilot points. Construction of a prior parameter uncertainty file which cites this matrix is easy. It is supplied as *param.unc*.

## 6.2 Posterior Parameter Uncertainties

The PREDUNC7 utility can be used to build a posterior parameter covariance matrix, together with a parameter uncertainty file that cites this matrix. Run PREDUNC7 as follows. (Note that the “reference variance” is the ratio of the objective function achieved through calibration to the number of non-zero weighted observations. This is 1.0 when observation weights are correct. Whenever a PEST control file is written by PWTADJ2, the answer to this question – whether it is posed by PREDUNC7 or any other PEST utility – is 1.0.)

|  |
| --- |
| **> predunc7**  PREDUNC7 Version 17.3. Watermark Numerical Computing.  Enter name of PEST control file: ***case3.pst***  Enter observation reference variance: ***1.0***  Enter name of prior parameter uncertainty file: ***param.unc***  Enter name for posterior parameter covariance matrix file: ***post\_cov.mat***  Enter name for posterior parameter uncertainty file: ***post\_param.unc***  Use which version of linear predictive uncertainty equation:-  if version optimized for small number of parameters - enter 1  if version optimized for small number of observations - enter 2  Enter your choice: ***1***  - reading PEST control file case3.pst....  - file case3.pst read ok.  - reading Jacobian matrix file case3.jco....  - file case3.jco read ok.  - reading parameter uncertainty file param.unc....  - covariance matrix file cov.mat read ok.  - parameter uncertainty file param.unc read ok.  - forming XtC-1(e)X matrix....  - inverting prior C(p) matrix....  - inverting [XtC-1(e)X + C-1(p)] matrix....  - writing file post\_cov.mat...  - file post\_cov.mat written ok.  - writing file post\_param.unc...  - file post\_param.unc written ok. |

The posterior parameter covariance matrix is stored in file *post\_cov.mat*. This is recorded in PEST matrix file format. As is usual for a covariance matrix, the variances of individual parameters are stored along the diagonal of this matrix; meanwhile off-diagonal matrix elements record covariances between pairs of parameters. We can extract the diagonal of this matrix using the MATDIAG utility which is supplied with PEST.

|  |
| --- |
| **> matdiag post\_cov.mat post\_cov\_diag.mat** |

Inspection of *post\_cov\_diag.mat* reveals parameter variances listed one-per-line in the first half of the file, and the names of parameters listed one-per-line in the second half of the file. (This is in accordance with PEST matrix file format, which is more appropriate for matrices than for vectors; recall, however, that a vector is a one-dimensional matrix.) With a little cutting and pasting using a text editor, parameter names and variances can be placed alongside each other. The file can then be imported into a spreadsheet program such as EXCEL in order to compute the square roots of variances, these being standard deviations. A map of posterior parameter standard deviations is provided in Figure 6.1. Recall that prior parameter standard deviations are all 0.447; these pertain to the log of K.



Figure 6.1. The spatial distribution of the posterior standard deviation of log K. Pilot points are also shown.

It is apparent from Figure 6.1 that the uncertainties of parameters near the top of the model domain are reduced very little through history matching against the twelve borehole heads. Uncertainty reduction of log K increases towards the south of the model domain. It is highest between observation wells.

## 6.3 Posterior Predictive Uncertainties

The PREDUNC1 utility can be used to explore the posterior uncertainties of the two predictions which are the focus of our attention. Recall that these are the easting of particle emergence from the south of the model domain, and the time of travel of the particle to the southern model boundary.

Before we run PREDUNC1 we must create two files. These files contain the sensitivities of the predictions of interest to all model parameters. These are recorded in two separate rows of the *case3.jco* Jacobian matrix. (Recall that, in the present case, our predictions are being “carried” through the calibration process as zero-weighted observations; therefore their sensitivities are available to us through the calibration JCO file. In other cases, a special “predictive JCO file” will need to be filled.)

Each row of a Jacobian matrix contains sensitivities of a particular model output to all model parameters. As an inspection of *case3.pst* reveals, our predictive model outputs are named *part\_time* and *part\_east*. A sensitivity vector for each of these predictions can be extracted from the *case3.jco* JCO file using the JROW2VEC utility. Run it twice as follows.

|  |
| --- |
| **> jrow2vec case3.jco part\_time part\_time.vec**  **> jrow2vec case3.jco part\_east part\_east.vec** |

After running JROW2VEC in this way, sensitivities of the particle travel time prediction to model parameters are stored in file *part\_time.vec*, while sensitivities of the particle emergence easting to model parameters are stored in file *part\_east.vec*. These are text files.

Now run PREDUNC1 twice to explore the posterior uncertainties of these predictions.

|  |
| --- |
| **> predunc1**    PREDUNC1 Version 17.3. Watermark Numerical Computing.  Enter name of PEST control file: ***case3.pst***  Enter observation reference variance: ***1.0***  Enter name of prior parameter uncertainty file: ***param.unc***  Enter name of predictive sensitivity matrix file: ***part\_time.vec***  Use which version of linear predictive uncertainty equation:-  if version optimized for small number of parameters - enter 1  if version optimized for small number of observations - enter 2  Enter your choice: ***1***  - reading PEST control file case3.pst....  - file case3.pst read ok.  - reading Jacobian matrix file case3.jco....  - file case3.jco read ok.  - reading predictive sensitivity matrix file part\_time.vec....  - file part\_time.vec read ok.  - reading parameter uncertainty file param.unc....  - covariance matrix file cov.mat read ok.  - parameter uncertainty file param.unc read ok.  - computing pre-calibration predictive uncertainty....  - forming XtC-1(e)X matrix....  - inverting C(p) matrix....  - inverting [XtC-1(e)X + C-1(p)] matrix....  - calculating post-calibration predictive uncertainty....  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* \*  \* Pre-cal predictive uncertainty = 1743.805 \*  \* Post-cal predictive uncertainty = 1622.987 \*  \* \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

|  |
| --- |
| **> predunc1**  PREDUNC1 Version 17.3. Watermark Numerical Computing.  Enter name of PEST control file: ***case3.pst***  Enter observation reference variance: ***1.0***  Enter name of prior parameter uncertainty file: ***param.unc***  Enter name of predictive sensitivity matrix file: ***part\_east.vec***  Use which version of linear predictive uncertainty equation:-  if version optimized for small number of parameters - enter 1  if version optimized for small number of observations - enter 2  Enter your choice: ***1***  - reading PEST control file case3.pst....  - file case3.pst read ok.  - reading Jacobian matrix file case3.jco....  - file case3.jco read ok.  - reading predictive sensitivity matrix file part\_east.vec....  - file part\_east.vec read ok.  - reading parameter uncertainty file param.unc....  - covariance matrix file cov.mat read ok.  - parameter uncertainty file param.unc read ok.  - computing pre-calibration predictive uncertainty....  - forming XtC-1(e)X matrix....  - inverting C(p) matrix....  - inverting [XtC-1(e)X + C-1(p)] matrix....  - calculating post-calibration predictive uncertainty....  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* \*  \* Pre-cal predictive uncertainty = 104.1928 \*  \* Post-cal predictive uncertainty = 36.80205 \*  \* \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

Table 6.1 summarises the outcomes of these two PREDUNC1 runs.

|  |  |  |
| --- | --- | --- |
| **Prediction** | **Prior uncertainty standard deviation** | **Posterior uncertainty standard deviation** |
| Particle travel time (days) | 1743.8 | 1623.0 |
| Easting of particle emergence (m) | 104.2 | 36.8 |

Table 6.1. Summary of PREDUNC1 uncertainty calculations.

It is apparent from the above table that calibration against twelve steady-state head observations reduces the uncertainty of the travel time prediction very little. However it does reduce the uncertainty of the easting-of-emergence prediction considerably. It is also apparent that the true particle travel time prediction lies approximately two posterior standard deviations away from that predicted by the calibrated model. It is thus included in the predictive uncertainty interval that PREDUNC1 assigns to this prediction. (The reality K field depicted in Figures 2.1 and 2.2 was purposely chosen to provide an extreme particle travel time prediction.) In contrast, the true value of the particle emergence easting is within one PREDUNC1-calculated standard deviation of the value calculated by the calibrated model.

The GENLINPRED utility (that is not demonstrated herein) can provide a breakdown of these figures. It uses the PREDUNC4 and PREDUNC5 utilities to examine:

* Pre- and post-calibration contributions to predictive uncertainty by different parameters;
* The information pertaining to these predictions carried by different observations; this is a measure of their ability to reduce the uncertainties of these predictions.

# 7. Sampling Parameter Distributions

## 7.1 General

A suite of utilities with the root name “RANDPAR” can be used to generate random realisations of parameters based on parameter stochasticity specified in a parameter uncertainty file. These utilities differ in the way that they store these random parameter sets. See Table 7.1. (Note that a JCB file is a binary file that is used for matrix storage by members of the PEST++ suite; it is also used by some members of the PEST suite.)

|  |  |
| --- | --- |
| **Program** | **Role** |
| RANDPAR | Generates random realisations of parameters based on the contents of a parameter uncertainty file. Parameter realisations are stored in a set of parameter value files. |
| RANDPAR1 | Same as RANDPAR, but slightly more efficient. |
| RANDPAR2 | Same as RANDPAR1; however parameter realisations are stored in a CSV file. |
| RANDPAR3 | Same as RANDPAR2; however parameter realisations are stored in a JCB file. |
| RANDPAR4 | Generates randomized parameter increments for use of PEST\_HP’s randomized Jacobian functionality. These are stored in a JCB file. |

Table 7.1. Members of the RANDPAR suite of programs.

In this section we will use RANDPAR1 to sample the prior parameter probability distribution, as well as the linear approximation to the posterior parameter probability distribution. We will then run the model multiple times using these parameter sets. There are a number of ways to undertake multiple model runs based on multiple parameters sets. They include the following.

* We could use of a batch loop in which model output files of interest are stored with a model run index built into their names. The contents of these files can then be read and collated using the RDMULRES utility. See documentation of RDMULRES in part 2 of the PEST manual. An example of such a batch loop is provided in RDMULRES documentation.
* PESTPP-SWP from the PEST++ suite can read parameter realisations that are stored in a CSV or JCB file. Hence the RANDPAR2 and RANDPAR3 utilities can be used to generate random parameter realisations for the use of this program. PESTPP-SWP can store model outputs of interest in the same types of files. Model runs based on different realisations can be conducted in parallel if desired.
* PEST\_HP can be run using the “/f” switch. When run in this way, PEST\_HP reads a succession of parameter value files. Model runs can be conducted in parallel if desired. Model run outcomes are stored in a “run results file”.

We will use PEST\_HP on this occasion. However the reader should be aware of the abovementioned alternative mechanisms for undertaking (possibly parallelised) model runs based on random parameter realisations.

## 7.2 Generating Samples

We will generate 50 samples of the prior parameter probability distribution, and then from the linear approximation to the posterior parameter probability distribution. Then we will calculate parameter means and standard deviations based on these samples.

To sample the prior, run RANDPAR1 as follows.

|  |
| --- |
| **> randpar1**  RANDPAR1 Version 17.3. Watermark Numerical Computing.  Enter name of existing PEST control file: ***case.pst***  - 104 parameters read from file case.pst.  - 104 of these are adjustable.  Use (log)normal or (log)uniform distrib for param generation? [n/u]: ***n***  Compute means as existing param values or range midpoints? [e/m]: ***e***  Respect parameter ranges? [y/n]: ***y***  Enter name of parameter uncertainty file: ***param.unc***  - covariance matrix file cov.mat read ok.  - parameter uncertainty file param.unc read ok.  Enter name of parameter ordering file (<Enter> if none): ***<Enter>***  Enter filename base for parameter value files: ***prior\_random***  How many of these files do you wish to generate? ***50***  Enter integer random number seed (<Enter> if default): ***<Enter>***  - undertaking singular value decomposition of covariance matrix...  - file prior\_random1.par written ok.  - file prior\_random2.par written ok.  etc |

Notice that the name of the PEST control file that was supplied to RANDPAR1 is *case.pst*. Recall that initial parameter values in this file are prior expected parameter values. Hence random realisations of prior parameter values should be centred on these.

We will now run RANDPAR1 again to sample the posterior parameter probability distribution. In this case the name of the PEST control file provided to RANDPAR1 is *case3.pst*. Thus random realisations will be centred on the posterior mean as evaluated by PEST when it calibrated the model; recall that calibrated parameter values are featured as “initial parameter values” in the *parameter data* section of *case3.pst*.

|  |
| --- |
| **> randpar1**  RANDPAR1 Version 17.3. Watermark Numerical Computing.  Enter name of existing PEST control file: ***case3.pst***  - 104 parameters read from file case3.pst.  - 104 of these are adjustable.  Use (log)normal or (log)uniform distrib for param generation? [n/u]: ***n***  Compute means as existing param values or range midpoints? [e/m]: ***e***  Respect parameter ranges? [y/n]: ***y***  Enter name of parameter uncertainty file: ***post\_param.unc***  - covariance matrix file post\_cov.mat read ok.  - parameter uncertainty file post\_param.unc read ok.  Enter name of parameter ordering file (<Enter> if none): ***<Enter>***  Enter filename base for parameter value files: ***post\_random***  How many of these files do you wish to generate? ***50***  Enter integer random number seed (<Enter> if default): ***<Enter>***  - undertaking singular value decomposition of covariance matrix...  - file post\_random1.par written ok.  - file post\_random2.par written ok.  etc |

## 7.3 Running the Model

PEST\_HP will now be run twice. The running of PEST\_HP requires that at least one parallelisation agent be operative. So start another command-line window, open to the current working folder (unless an extra command-line window remains open from the previous PEST\_HP run). In one of the command-line windows type the command:

|  |
| --- |
| **> agent\_hp case /h %computername%:4004** |

In the other command-line window, run PEST\_HP as follows. (Be sure to leave a space between “/h” and “:4004”.)

|  |
| --- |
| **> pest\_hp case /f /h :4004**  PEST\_HP Version 17.42. Watermark Numerical Computing.  PEST\_HP has been called with the "/f" switch.  PEST\_HP will run the model repeatedly, using parameter values recorded  in a sequence of parameter value files. It will record all model-calculated  observations in a run results file.  Enter filename base of parameter value files: ***prior\_random***  Enter first index to use: ***1***  Enter last index to use: ***50***  Enter parallel run packet size: ***5***  Enter name for run results file: ***prior.rrf*** |

PEST\_HP undertakes 5 model runs at a time. After each packet of 5 runs, it writes the objective function to the screen. These objective functions are high, this being an outcome of the fact that the parameter value files which it reads are sampled from the prior parameter probability distribution. These objective functions are also recorded in the run record file *case.rec*. Meanwwhile, parameter values and model output values corresponding to all realisations are recorded in the run results file *prior.rrf*. Recall that, in the present case, model output values include both model outputs that are used in the calibration process and model predictions in which we are interested.

Now run PEST\_HP again based on samples of the linearised posterior parameter probability distribution. Start up the agent as follows.

|  |
| --- |
| **> agent\_hp case3 /h %computername%:4004** |

And then, in the other window, run PEST\_HP as follows:

|  |
| --- |
| **> pest\_hp case3 /f /h :4004**  PEST\_HP Version 17.42. Watermark Numerical Computing.  PEST\_HP has been called with the "/f" switch.  PEST\_HP will run the model repeatedly, using parameter values recorded  in a sequence of parameter value files. It will record all model-calculated  observations in a run results file.  Enter filename base of parameter value files: ***post\_random***  Enter first index to use: ***1***  Enter last index to use: ***50***  Enter parallel run packet size: ***5***  Enter name for run results file: ***posterior.rrf*** |

The screen display shows that objective functions are much lower.

A number of PEST utilities read and process the contents of a run results file (i.e. a RRF file). For example, RRFCALCPSI can re-calculate objective functions based on a different set of weights without having to run the model again, while RRF2JCO can compute an approximation to a Jacobian matrix based on parameter-to-observation covariances that are calculated from the contents of this file.

In the present instance we will transfer the contents of the run results files to CSV files. Two CSV files are required to hold the contents of a single RRF file. One of these stores parameter values while the other stores model output values. Run RRF2CSV twice as follows.

|  |
| --- |
| **> rrf2csv prior.rrf prior\_param.csv prior\_output.csv**  **> rrf2csv posterior.rrf posterior\_param.csv posterior\_output.csv** |

Files *prior\_output.csv* and *posterior\_output.csv* can now be imported into a spreadsheet package such as EXCEL for analysis. Focusing on the two predictions, the following results are obtained.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prediction** | **Prior mean** | **Prior standard deviation** | **Posterior mean** | **Posterior standard deviation** |
| part\_time (days) | 6704.7 | 1498.1 | 6822.5 | 1217.6 |
| part\_east (m) | 248.1 | 73.3 | 235.8 | 42.0 |

Table 7.1. Predictive uncertainties estimated from 100 model runs, 50 of which were based on prior parameter realisations and 50 of which were based on posterior parameter realisations.

A comparison with Table 6.1 reveals that the outcomes of this analysis are not too different from those of linear analysis. Nevertheless there are differences. These differences may be due to one or more of the following factors.

1. Numbers provided in Table 7.1 are based on a sample size of only 50.
2. Linear analysis (especially of prior predictive uncertainties) is only approximate. The numbers that are presented in Table 6.1 can therefore be viewed as only representative.
3. Estimates of posterior predictive uncertainties that are presented in Table 7.1 are based on samples of an approximate, linearized, posterior parameter probability distribution.

## 7.4 Some Pictures

Figure 7.1 shows pictures of the first two prior parameter fields generated by RANDPAR1, together with the particle track and heads pertaining to each. Figure 7.2 does the same for the first two samples of the linearised posterior parameter probability distribution.

 

Figure 7.1. Two samples of the prior probability distribution of hydraulic conductivity, together with respective heads and particle paths.

 

Figure 7.2. Two samples of the posterior probability distribution of hydraulic conductivity, together with respective heads and particle paths.

# 8. Sampling the Posterior using PESTPP-IES

## 8.1 General

Use of the PESTPP-IES ensemble smoother allows a modeller to avoid calibration altogether. Instead, samples of the prior parameter probability distribution can be morphed into samples of the posterior parameter probability distribution. This realisation alteration process is numerically efficient, requiring only as many model runs per iteration as the number of parameter realisations that are undergoing alteration (plus a few more for Marquardt lambda testing).

## 8.2 Running PESTPP-IES

We will modify *case3.pst* so that it is ready for use with PESTPP-IES. First:

|  |
| --- |
| **> copy case.pst case4.pst** |

Now add the following lines to the bottom of *case4.pst*.

|  |
| --- |
| ++ ies\_num\_reals(50)  ++ parcov(param.unc)  ++ ies\_subset\_size(2)  ++ ies\_autoadaloc(true) |

The above lines instruct PESTPP-IES to do the following:

* Use 50 realisations;
* Look to file *param.unc* for statistical characterisation of prior parameter uncertainty;
* Randomly select two sets of realisations during each iteration of the optimisation process for Marquardt lambda testing;
* Implement auto localisation.

Set NOPTMAX (first variable on the 9th line of *case4.pst*) to 5, so that PESTPP-IES stops after 5 iterations. Now run PESTPP-IES using the command:

|  |
| --- |
| **> pestpp-ies case4** |

(Note that PESTPP-IES can also undertake runs in parallel. Its protocol for undertaking parallel model runs is the same of that of PEST\_HP, except for the fact that the agent is, in fact, PESTPP-IES itself.)

## 8.3 Inspecting the Results

PESTPP-IES records a number of CSV files. In the present case, because there are not too many parameters and observations, these CSV files are easily imported into packages such as EXCEL for viewing, processing and plotting.

File *case4.phi.actual.csv* records the objective function associated with each realisation as the history-matching process progresses. These are graphed in Figure 8.1.



Figure 8.1. Objective function vs iteration number for all ensemble realisations. Note the logarithmic y-axis scale.

Model outputs are recorded in file *case4.N.obs.csv* where *N* is the iteration number. Minor processing of data that is recorded in files *case4.0.obs.csv* and *case4.5.obs.csv* allows filling of the entries of Table 8.1. These compare well with respective entries of Table 7.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prediction** | **Prior mean** | **Prior standard deviation** | **Posterior mean** | **Posterior standard deviation** |
| part\_time (days) | 6220.2 | 1674.5 | 6254.0 | 1132.1 |
| part\_east (m) | 250.7 | 70.4 | 227.4 | 47.1 |

Table 8.1. Prior and posterior statistics for the two model predictions of interest.

## 8.4 Extracting Results

Individual realisations can be extracted from CSV files produced by PESTPP-IES using the CSV2PAR utility supplied with PEST. Parameter sets comprising these realisations are recorded in parameter value files (i.e. PAR files). To extract parameter values corresponding to realisation 0 from *case4.5.par.csv*, run CSV2PAR, responding to its prompts as follows:

|  |
| --- |
| **> csv2par**  CSV2PAR Version 17.3. Watermark Numerical Computing.  Enter name of CSV file: ***case4.5.par.csv***  Extract param vals from a row or column of this file? [r/c]: ***r***  Enter name of row: ***0***  Enter name for parameter value file: ***case4\_r0.par***  - file case4.5.par.csv read ok.  - file case4\_r0.par written ok. |

We can now undertake a model run based on these parameters by first constructing a PEST control file in which these parameters are featured as initial values, and by then running PEST with NOPTMAX set to zero in this file.

|  |
| --- |
| **> parrep case4\_r0.par case.pst temp.pst** |

After setting NOPTMAX to 0 in *temp.pst* (this is the first variable on the 9th line of this file), run PEST as follows.

|  |
| --- |
| **> pest temp** |

As expected, the objective function is quite low – about 10.4. Hence the parameter field is a sample of the posterior parameter probability distribution. The parameter field for realisation 0, as well as heads and particle path that are calculated using this parameter field, are shown in Figure 8.2, along with those corresponding to realisation 1.

 

Figure 8.2. Two samples of the posterior probability distribution of hydraulic conductivity, together with heads and particle paths.

Figure 8.3 shows results for the so-called “base realisation”. This is not like other realisations. All realisations used in the ensemble smoother process, except for the base realisation, are modified from random samples of the prior parameter probability distribution. However the base realisation is modified from the (uniform) set of initial values that are featured in the PEST control file (in this case *case.pst*). Recall that these are prior expected parameter values. Ideally, this history-matched realisation should resemble the parameter field that is achieved through model calibration. A comparison with Figure 4.1 shows that the two parameter fields do bear some similarities. However that attained through the ensemble smoother process is “rougher” than that attained through traditional calibration.



Figure 8.3. The history-match-adjusted “base realization” of hydraulic conductivity, together with heads and particle path.

# 9. Data Space Inversion

## 9.1 General

Data-space inversion explores the posterior uncertainty of a prediction of interest using a comparatively small number of model runs. While it allows constraints to be imposed on model predictions by the historical behaviour of a system, it does not apply these constraints to parameters. Instead, it by-passes parameter adjustment, focussing only on the link between historical system behaviour and future system behaviour as calculated by the model.

In the present case, we will base our assessment of history-match-constrained predictive uncertainty on the 50 model runs that PESTPP-IES undertook during its zeroth iteration. Model runs that are undertaken during this iteration employ parameter fields that are sampled from the prior parameter probability distribution.

## 9.2 Running DSI1

Run the DSI1 program (supplied with the PEST suite), responding to its prompts as follows in order to obtain estimates of the prior and posterior predictive uncertainties of the *part\_time* prediction.

|  |
| --- |
| **> dsi1**  DSI1 Version 17.3. Watermark Numerical Computing.  Enter seed for random number generator (<Enter> if 1111): ***<Enter>***  Enter name of PEST control file: ***case.pst***  - 14 observations read from file case.pst.  - 12 of these have a nonzero weight.  Enter name of transformation file (<Enter> if none): ***<Enter>***  Enter name of CSV file containing model-generated observations: ***case4.0.obs.csv***  How many realizations to read from this file? ***50***  - file case4.0.obs.csv read ok.  Enter name of CSV file holding 50 prediction realizations: ***case4.0.obs.csv***  Enter name of prediction: ***part\_time***  - 50 predictive realizations read from file case4.0.obs.csv.  Transform prediction? [y/n]: ***n***  Work in normal score transform space? [y/n]: ***n***  Enter name for output file: ***dsi1\_part\_time.dat***  Enter "energy threshold" for SVD truncation.  (Normally between 0.9 and 0.99): ***0.99***  - computing ZtZ matrix...  - performing singular value decomposition of ZtZ matrix...  - formulation of US(-1) matrix...  - computing posterior stats...  - recording output file...  - file dsi\_part\_time.dat written ok. |

Now run it again to explore the prior and posterior uncertainties of the *part\_east* prediction.

|  |
| --- |
| **> dsi1**  DSI1 Version 17.3. Watermark Numerical Computing.  Enter seed for random number generator (<Enter> if 1111): ***<Enter>***  Enter name of PEST control file: ***case.pst***  - 14 observations read from file case.pst.  - 12 of these have a nonzero weight.  Enter name of transformation file (<Enter> if none): ***<Enter>***  Enter name of CSV file containing model-generated observations: ***case4.0.obs.csv***  How many realizations to read from this file? ***50***  - file case4.0.obs.csv read ok.  Enter name of CSV file holding 50 prediction realizations: ***case4.0.obs.csv***  Enter name of prediction: ***part\_east***  - 50 predictive realizations read from file case4.0.obs.csv.  Transform prediction? [y/n]: ***n***  Work in normal score transform space? [y/n]: ***n***  Enter name for output file: ***dsi1\_part\_east.dat***  Enter "energy threshold" for SVD truncation.  (Normally between 0.9 and 0.99): ***0.99***  - computing ZtZ matrix...  - performing singular value decomposition of ZtZ matrix...  - formulation of US(-1) matrix...  - computing posterior stats...  - recording output file...  - file dsi1\_part\_east.dat written ok. |

## 9.2 DSI results

Results are presented in Table 9.1. Standard deviations that are listed in this table are calculated as half the difference between “high\_68” and “low\_68” in tables headed “predictive confidence limits” on respective DSI1 output files (i.e. *dsi1\_part\_time.dat* and *dsi1\_part\_east.dat*).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prediction** | **Prior mean** | **Prior standard deviation** | **Posterior mean** | **Posterior standard deviation** |
| part\_time (days) | 6220.2 | 1674.5 | 6167.8 | 1465.0 |
| part\_east (metres) | 250.7 | 70.4 | 244.6 | 47.7 |

Table 9.1. Prior and posterior statistics for the two model predictions of interest.

The results that are listed in Table 9.1 are similar to those provided in previous tables. However, these results are based on only 50 model runs!

# 10. Conclusions

This workshop has demonstrated four types of uncertainty analysis. These are:

* Calibration followed by linear analysis;
* Monte-Carlo analysis based on a linearised posterior (i.e. pseudo-nonlinear analysis);
* Monte-Carlo analysis based on a posterior parameter ensemble computed by PESTPP-IES;
* Data space inversion.

The outcomes of all of these analyses for the two predictions that are the focus of the present study are presented in Tables 10.1 and 10.2. For linear analysis, the prior mean that is presented in these tables is the prediction made using prior expected parameter values. These values are listed in file *case.pst*; they are all 1 m/day. Meanwhile the posterior mean prediction for linear analysis is that calculated by the calibrated model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Prior mean** | **Prior standard deviation** | **Posterior mean** | **Posterior standard deviation** |
| Linear | 6850.0 | 1743.8 | 6840.8 | 1623.0 |
| Pseudo nonlinear | 6704.7 | 1498.1 | 6822.5 | 1217.6 |
| PESTPP-IES | 6220.2 | 1674.5 | 6254.0 | 1132.1 |
| Data space inversion | 6220.2 | 1674.5 | 6167.8 | 1465.0 |

Table 10.1. Prior and posterior statistics for the particle travel time. The units of all entities are days.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Prior mean** | **Prior standard deviation** | **Posterior mean** | **Posterior standard deviation** |
| Linear | 245.0 | 104.2 | 225.9 | 36.8 |
| Pseudo nonlinear | 248.1 | 73.3 | 235.8 | 42.0 |
| PESTPP-IES | 250.7 | 70.4 | 227.4 | 47.1 |
| Data space inversion | 250.7 | 70.4 | 244.6 | 47.7 |

Table 10.2. Prior and posterior statistics for the particle emergence easting. The units of all entities are metres.

Prior and posterior uncertainties calculated through all of these analyses are approximate in one way or another. The limitations of linear analysis are obvious. Analyses based on IES and DSI are hampered by the limited sample size. Nevertheless, the similarity of results is gratifying.

In real-world contexts, all methods except data space inversion suffer a further approximation arising from the assumption of a multi-Gaussian prior probability distribution of model parameters. All of the above methods assume multi-Gaussian measurement noise.