9. Randomized Jacobian

9.1 Introduction

This section describes functionality included in PEST_HP that often allows progress to be made with a high level of model run efficiency in difficult parameter estimation contexts, especially those in which parameter numbers are large and where the dimensionality of the calibration solution space is small. This progress may come at a price, however. While the methodology described herein may allow a good fit to be attained between model outputs and a calibration dataset, the solution to the inverse problem that is thereby achieved may not be that of minimized error variance. However, if proper precautions are taken, the solution achieved through the methodology described herein may not deviate too far from this optimal solution.

Deviations from inverse solution optimality can arise where parameter upgrade directions are not confined to the calibration solution space. Consequently, null space components are introduced to parameters as they are upgraded. These are not desirable features of the so-called "calibrated parameter field". However this is the price to be paid for the speed with which a good fit can be attained with a calibration dataset where parameter upgrades are based on a randomized Jacobian matrix. Furthermore, if steps are taken to keep such "null space entrainment" to a minimum, the calibrated parameter field may still be quite acceptable.

Because of its model run efficiency, the methodology described herein may greatly reduce the numerical burden of direct predictive hypothesis testing, whereby an hypothesized prediction is included in the calibration dataset. Model run efficiency is particularly important under these circumstances as a model must simulate both past and future system states.

9.2 Overview

In its normal operation, PEST_HP approximates differentials comprising a Jacobian matrix using finite parameter differences. Under these circumstances, filling of the Jacobian matrix requires at least as many model runs as there are adjustable parameters. Where an inverse problem is ill-posed, the Jacobian matrix can then be processed using a subspace method such as singular value decomposition (SVD). SVD reduces the dimensionality of an inverse problem by selecting combinations of parameters for estimation in place of individual parameters. In doing this, it notionally formulates a full-rank Jacobian matrix from the original, rank-deficient Jacobian matrix; this full rank Jacobian matrix complements the solution subspace.

Methods such as randomized SVD have received a considerable amount of attention in the recent numerical methods literature; see, for example, Halko et al (2011) for a review. They are closely related to the use of ensemble methods as a solution device for ill-posed inverse problems. Ensemble methods have been used for both calibration and calibration-constrained uncertainty analysis; see, for example, White (2018) and Chada et al (2018). The utility of methods based on random realizations of parameters rests on the premise that a Jacobian matrix that is employed in solution of an ill-posed inverse problem need only have a rank that is as large as that of the matrix which SVD will ultimately derive to estimate values for a limited number of estimable parameter combinations. The rank of this matrix is equal to the number of parameter combinations which are thus estimated (i.e. to the dimensionality of the calibration solution space). Construction of a

Jacobian matrix which is rank-sufficient in terms of the dimensionality of the calibration solution space, while being rank deficient in terms of the number of adjustable parameters, can be achieved if random combinations of parameters are employed to explore the range space of this matrix. The dimensionality of its range space is equal to the number of parameter combinations which can be estimated, and hence to the dimensionality of the solution space. The number of sampled parameter sets that are required to explore this space and define its dimensions need only be slightly greater than the dimensionality of the solution space.

When employing PEST_HP's randomized Jacobian functionality, random parameter sets can be generated by PEST_HP itself. Alternatively, they can be provided by the user. Both of these means of random parameter set generation can be combined in a single PEST_HP run. If random parameter sets are generated by PEST_HP, every parameter is considered to be statistically independent of every other parameter; the characteristics of random parameter sets are inherited from the variables which would otherwise govern parameter variation for the purpose of finite-difference derivatives calculation. Alternatively, user-supplied random parameter sets can be generated using whatever specifications of parameter randomness that a user sees fit. Furthermore, PEST_HP can alter its source of random parameter sets from iteration to iteration of the inversion process – either randomly, or according to a user-specified strategy.

Regardless of the source of random parameter vectors, the deleterious effects of using a rank-deficient Jacobian matrix can be mitigated through deployment of an appropriate "localization" strategy. This can dampen the occurrence of phantom parameter-to-observation sensitivities arising from the use of an insufficient number of random parameter increments. Localization can be implemented automatically by PEST_HP; alternatively, or as well, a user can implement it him/herself by supplying his/her own localization matrix.

As stated above, the number of random parameter sets for which model runs are carried out in order to construct a Jacobian matrix of sufficient rank to allow model outputs to fit a calibration dataset can be considerably less than the actual number of parameters that are awarded to a model. Naturally, PEST_HP parallelizes these model runs. Once a rank-deficient approximation to a Jacobian matrix has been calculated using these random parameter set realizations, solution of the inverse problem can then be undertaken using any of the methods normally supported by PEST_HP. Singular value decomposition with SVDMODE set to 2 is often recommended as PEST_HP's use of different Marquardt lambdas when testing parameter upgrades is then able to achieve singular value compression, the benefits of which are described by Engel et al (2017).

As usual, it is recommended that the inverse problem solution process includes Tikhonov regularisation as a device for maximizing parameter reasonableness. As is discussed elsewhere in this manual, the numerical burden of Tikhonov regularization can be reduced considerably through selection of the "high-speed regularization" option accessed through the REG2MEASRAT control variable in the "regularisation" section of a PEST control file.

9.3 Randomized Jacobian Matrix: Theory

9.3.1 Calculating the Jacobian Matrix

If a model is linear, its operation can be represented by a matrix. Let this matrix be designated as **J** when the model is run under calibration conditions. Let us represent model outputs that correspond

to members of a calibration dataset by the vector **o**. Let the vector **k** represent model parameters. Then

$$\mathbf{o} = \mathbf{J}\mathbf{k} \tag{9.1}$$

Suppose that elements of **k** are randomly generated based on a covariance matrix D(**k**) (which we distinguish from C(**k**), which is often used in PEST and related documentation to denote the prior parameter covariance matrix). Then the joint covariance matrix of **o** and **k** can be computed as

$$\mathrm{C}\left(\begin{bmatrix}\mathbf{o}\\ \mathbf{k}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{J}\\ \mathbf{I}\end{bmatrix} \mathrm{D}(\mathbf{k}) [\mathbf{J}^{\mathrm{t}} \quad \mathbf{I}] \tag{9.2}$$

After carrying out the matrix products implied by equation 9.2, and extracting the pertinent submatrix from the resulting full matrix, we find

$$\mathrm{C}(\mathbf{o},\mathbf{k}) = \mathbf{J}\mathrm{D}(\mathbf{k}) \tag{9.3}$$

where C(**o**,**k**) expresses covariances between the elements of **o** and those of **k**. From (9.3),

$$\mathbf{J} = \mathrm{C}(\mathbf{o},\mathbf{k})\mathrm{D}^{-1}(\mathbf{k}) \tag{9.4}$$

For a nonlinear model, the Jacobian matrix **J** constitutes a local linearization of the action of the model on the set of parameters achieved at a certain stage of the inversion process; we denote this set of parameters as $\underline{\mathbf{k}}$.

An empirical estimate of C(**o**,**k**) can be obtained by running the model using different random realizations of **k** centred on $\underline{\mathbf{k}}$. The covariance between the *i*'th element of **o**, o_i, and the *j*'th element of **k**, k_j, can then be calculated as

$$\mathrm{C}(\mathbf{o},\mathbf{k})_{i,j} = \frac{\sum(o_i - \underline{o_i})(k_j - \underline{k_j})}{N-1} \tag{9.5}$$

where the summation is undertaken across all random parameter sets. In equation 9.5 $\underline{k_j}$ is the mean value of parameter k_j over all realizations while $\underline{o_i}$ is the mean value of model output o_i over all realizations.

For random parameter increments generated by PEST_HP, D(**k**) is a diagonal matrix. Each element of the diagonal is the variance (square of standard deviation) of the respective, locally randomized, parameter. This is because, in calculation of the covariance matrix featured in equation 9.3, PEST_HP considers individual parameters comprising elements of the vector **k** to be statistically independent of each other. Furthermore, parameter standard deviations are generally small. (However larger standard deviations can be employed if desired.) These will generally be of similar magnitude to the increments that would otherwise be used for calculation of a Jacobian matrix using finite parameter differences. The use of small standard deviations serves two purposes.

1. Elements of the matrix **J** (the local Jacobian matrix) calculated using equation 9.4 are a better approximation to true derivatives than if parameter randomization was based on larger standard deviations; hopefully, this accelerates solution convergence of the inverse problem.
2. For some models (for example geothermal reservoir models run under natural state conditions, and groundwater models run under steady state conditions), execution speed

can be dramatically increased if initial system states supplied to the simulator's solver approximate the actual simulated system states. PEST_HP can provide these initial states to the model from previously-calculated solution states, and update them during every iteration of the inversion process, using its file distribution functionality. However system state solutions determined during a previous iteration of the inversion process can only serve as useful initial system state conditions for the next iteration of that process if random parameter variations from their current reference values are restricted in size.

Where random parameter increments are provided by a user, D(**k**) may not be diagonal. For example, random parameter increments may have been generated using a covariance matrix that is proportional to the prior parameter covariance matrix, C(**k**). This is the user's choice. PEST_HP does, however, expect that random parameter increments have a mean of zero for non-log-transformed parameters and a mean of one for log-transformed parameters. Randomized increments are added to current parameter values in the former case and multiply current parameter values in the latter case. (If random parameter increment sets are generated using the RANDPAR4 utility supplied with the PEST suite, this will be achieved automatically.)

Let the matrix **K** denote a user-supplied parameter increment ensemble. This matrix has as many rows as there are adjustable parameters and as many columns as there are parameter increment realizations. PEST_HP calculates an approximate covariance matrix D(**k**) from **K** as

$$\mathrm{D}(\mathbf{k}) = \frac{\mathbf{K}^t\mathbf{K}}{N-1} \tag{9.6}$$

Because D(**k**) is rank-deficient if the number of parameters is greater than the number of parameter increment realizations (which is generally the case), it cannot be inverted in accordance with the requirements of equation 9.4. Furthermore, if it were of full rank (this requiring at least as many parameter increment realizations as there are adjustable parameters), inversion of this matrix would incur a high numerical burden in highly-parameterized settings. PEST_HP overcomes the first of these problems by first subjecting the matrix **K** to singular value decomposition as

$$\mathbf{K} = \mathbf{USV}^t \tag{9.7}$$

and then calculating the pseudoinverse $\mathbf{K}^t\mathbf{K}^-$ of $\mathbf{K}^t\mathbf{K}$ as

$$\mathbf{K}^t\mathbf{K}^- = \mathbf{US}^{-2}\mathbf{U}^t \tag{9.8}$$

This methodology for calculating an approximation to **J** is not unlike that described by Chen and Oliver (2013) to support their Jacobian-enhanced, ensemble Kalman smoother scheme. However in their case parameter randomization plays two roles. Firstly, it is employed to create parameter fields which constitute an evolving ensemble which is used to quantify parameter and predictive uncertainty. Secondly, the outcomes of model runs conducted using these parameter fields are employed in equations that are not unlike those discussed above to compute an approximation to the Jacobian matrix on which modification of these parameter fields is based. An important difference between the Chen and Oliver randomization scheme and that described in the present document is that parameter variances employed in the present scheme can (and probably should) be much smaller than those used by Chen and Oliver. This is because, in the present case, randomness applies to parameter increments rather than to parameters themselves. These increments are generated for use in model runs that are dedicated solely to construction of an approximate Jacobian matrix. The latter is then used to adjust a single parameter field to better satisfy calibration constraints. Another important difference is that, with appropriate selection of pertinent control variables, the rank-deficiency of the Jacobian matrix computed by PEST_HP

decreases with progress of the inversion process. The Jacobian matrix that is achieved at the end of this process may therefore constitute a reasonable approximation to the true parameter Jacobian matrix. As such, it may form the basis for approximate linear parameter uncertainty and identifiability analysis.

As has already been mentioned, the method described herein also bears some resemblance to randomization algorithms which are used for singular value decomposition of large matrices. A major difference between these methods and that described herein however is that randomization is used to build an approximation to the Jacobian matrix herein, rather than to decompose an existing Jacobian matrix. However if the model to which PEST_HP is linked possesses an adjoint solver, so that "backward" model runs can be undertaken to directly compute matrix multiplications involving $\mathbf{J}^t$, the distinction between these two roles can fade. Algorithms described by Halko et al (2011) and Tropp et al (2016) could then be employed for rapid calculation of upgrade vectors without the subspace misalignment problems that attend the present methodology.

9.3.2 Localization

"Localization" is a term borrowed from ensemble Kalman filter and smoother terminology. It describes a process through which errors in a Jacobian matrix incurred through the use of random parameter increments are mitigated. At the same time, the degree of rank-deficiency of the Jacobian matrix can be reduced. Localization reduces the values of randomized Jacobian matrix elements (possibly to zero) if it is possible that these values may, in fact, be random noise incurred through the randomized Jacobian matrix construction process. This strategy is referred to as "localization" because sometimes a modeller may know that one parameter cannot influence a particular model output because the latter is far removed in space from the former; in this case he/she is entitled to replace a spurious sensitivity with the known value of zero.

PEST_HP allows a user to supply a "localization matrix". Let us refer to this matrix as **L**. It must have the same dimensions as the Jacobian matrix **J**. Then, on each occasion that PEST_HP calculates a Jacobian matrix using equation 9.4, it replaces it with a new Jacobian matrix **J**' that is calculated using the equation

$$\mathbf{J}' = \mathbf{J} \circ \mathbf{L} \tag{9.9}$$

In equation 9.9, "$\circ$" represents the Schur matrix product; this denotes multiplication of corresponding elements of each matrix on the right side of the equation to form the matrix on the left side of the equation. For **L** to be a valid localization matrix, each of its elements must lie in the range [0,1].

If asked to do so, PEST_HP can compute its own localization matrix (and re-compute it during each iteration of the inversion process) using an "adaptive localization" methodology that is similar to that described by Luo and Bhakta (2020). However instead of applying this localization matrix to the Jacobian matrix, PEST_HP applies it to the parameter-to-observation covariance matrix C(**o**,**k**) computed using equation 9.5. (This has the same dimensions as the Jacobian matrix.) First PEST_HP computes a matrix of correlation coefficients between observations and parameters. The empirical correlation coefficient ρ_{ij} between observation *i* and parameter *j* is computed as

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_i^2 \sigma_j^2}} \tag{9.10}$$

where

σ_{ij} is the empirical covariance between observation *i* and parameter *j* (i.e. element (*i*,*j*) of C(**o**,**k**));

σ^2_j is the empirical variance of parameter *j* (i.e. the j'th diagonal element of D(**k**) of equation 9.6); and

σ^2_j is the empirical variance of observation *i*.

Where there is no correlation between observation *i* and parameter *j*, then the empirical value of ρ_{ij} calculated as above may nevertheless possess a non-zero value. In fact, its expected absolute value is $N^{1/2}$ where *N* is the number of random parameter sets used in construction of the empirical covariance matrices C(**o**,**k**) and D(**k**). If asked to do so, PEST_HP compares the absolute value of each correlation coefficient ρ_{ij} with t_α computed as

$$t_\alpha = \frac{|\rho_{ij}|}{N^{½}} \times 2\ln(N) \times A \qquad (9.11)$$

Where *A* is the user-supplied value of AUTOTHRESH (see below). If $|\rho_{ij}|$ is less than t_α, the value of σ_{ij} (i.e. $C(\mathbf{o},\mathbf{k})_{ij}$) is reduced, as this covariance may actually be spurious rather than real as an outcome of the fact that *N* is finite. If "hard autolocalization" is requested by a user, then σ_{ij} is set to zero. If "soft localization" is requested by the user, then σ_{ij} is multiplied by a value between zero and one using the Gaspari-Cohn tapering function; see Luo and Bhakta (2020) for more details (including a description of this function). The value of this tapering function is 1.0 when $|\rho_{ij}|$ is greater than or equal to t_α. It diminishes smoothly to zero as $|\rho_{ij}|$ falls to zero.

A value of between 0.1 and 1.0 is often appropriate for AUTOTHRESH. The use of this autolocalization threshold value is based on the premise that empirical correlation coefficients of less than t_α may be indicative of random noise, and hence may not be real. The corresponding element of C(**o**,**k**) is therefore replaced with a value of zero, or with a value of diminished magnitude. Hence only "real", and not spurious, parameter-to-observation correlations remain in C(**o**,**k**) and are therefore used to calculate **J**.

At the end of each iteration of an inversion process in which autolocalization is implemented, PEST_HP appends information to an "autolocalization report file". This file is named *case.alr*, where *case* is the filename base of the PEST control file. The autolocalization report file contains three columns of information. The first column specifies the iteration number. The second column specifies the number of elements contained in the C(**o**,**k**) matrix; this does not change from iteration to iteration. The third column specifies the number of elements within this matrix for which corresponding parameter-to-observation correlation coefficients are less than t_α. Corresponding elements of C(**o**,**k**) are either set to zero (for hard autolocalization) or diminished in value (for soft autolocalization). An inspection of the autolocalization report file may assist a user in choosing an appropriate value for t_α. The number of elements of C(**o**,**k**) whose values are diminished is also reported to the screen.

9.4 Randomized Jacobian Matrix: Practice

9.4.1 General

As has already been discussed, PEST_HP can generate random parameter increments itself. Alternatively, it can accept random parameter increments generated by another program. Each of these alternatives is discussed below.

Regardless of how they are generated, a modeller is faced with the choice of how many random parameter sets *N* to use in building a diminished rank approximation to the Jacobian matrix. There are two factors at play here. The larger is *N*, the better will the empirical covariance matrix computed using equation 9.4 approximate the true covariance matrix. This is because spurious parameter-to-observation correlations are reduced as *N* increases; as has already been discussed, they are reduced in proportion to $N^{½}$. On the other hand, the whole purpose behind construction of a randomized Jacobian matrix is to reduce the numerical burden of parameter estimation. Therefore, *N* should ideally be significantly smaller than the number of adjustable parameters. In practice, *N* must be at least as large as the dimensionality of the calibration solution space; theoretically, it should be slightly larger than this, as a small amount of parameter redundancy provides a better guarantee that the dimensionality of the range space of **J** is properly defined by model outcomes calculated using these increments.

It is up to the modeller to choose a value for *N* that reconciles these two conflicting demands. The problem of choosing *N* is exacerbated by the fact that the dimensionality of the calibration solution space (and hence the Jacobian matrix range space) may not be known in advance. PEST_HP accommodates uncertainty in choosing a suitable value for *N* by allowing it to increase, on an as-needed basis, as the inversion process progresses.

9.4.2 PEST_HP-Generated Parameter Increments

PEST_HP employs a normal probability distribution to generate random realizations of parameters on which to base model runs to fill the Jacobian matrix. The mean value of each parameter is its current reference value (i.e. the best parameter value inherited from the parameter upgrade process conducted during the previous iteration). Hence the mean value of parameter increments is zero. Parameter increment standard deviations are based on variables that would otherwise be used to control finite-difference derivatives calculation.

Ideally, standard deviations employed in generation of random parameter increment values should be small. However if they are too small, then round-off errors can diminish the integrity of variances and covariances calculated using equations 9.5 and 9.6. Numerical errors in C(**o**,**k**) can be exacerbated by model solver convergence difficulties; the resulting "model output noise" can impair the numerical integrity of small differences of large quantities. In generating parameter realizations, PEST_HP establishes the standard deviation for each parameter as the increment that the parameter would have if two-point (rather than three or five point) finite-difference derivatives were employed to fill the Jacobian matrix. The PEST control variables which determine these increments pertain to parameter groups, and are thus recorded in the "parameter groups" section of a PEST control file. These are the DERINC, DERINCLB, and INCTYP variables. However PEST_HP allows parameter standard deviations calculated in this way to be multiplied by a user-specified factor that is applied to all parameter groups. Hence they can be made as large or as small as a user desires. Alternatively,

values for DERINC, DERINCLB and INCTYP can be chosen specifically to achieve desired standard deviations.

When calculating randomly-incremented values for parameters, PEST_HP ensures that these values do not transgress parameter bounds. If, at a particular stage of the inversion process, the current reference value for a parameter is at its upper or lower bound, the sign of the random parameter increment is reversed if necessary to ensure that the parameter's increment moves its value away from this bound rather than across it. On the other hand, if a parameter is not currently at its bound, but its incremented value lies above the parameter's upper bound or below its lower bound, then the value ascribed to the randomly-incremented parameter is adjusted to coincide with the bound.

9.4.3 User-Supplied Parameter Increments

PEST_HP can be directed to one or a number of so-called "JCB files" to read parameter increment realizations. These are binary files that are produced by PEST-suite programs such as RANDPAR3 and RANDPAR4. Their contents can be examined and manipulated by other PEST-suite programs such as JCB2CSV, JCB2PAR and JCB2RRF.

Certain requirements must be met by a random parameter increment JCB file that is supplied to PEST_HP. PEST_HP can check for some of these, but not for all of them.

First, every parameter that is declared as adjustable in the PEST control file on which the PEST_HP inversion process is based must be featured in the user-supplied JCB file. However, there is no need to supply realizations of fixed or tied parameters in these JCB files. If values for these parameters are in fact supplied, they are ignored by PEST_HP, for fixed parameters are not incremented, and tied parameters are calculated from the values of their parent parameters.

Second, as stated above, because the realizations that are supplied in a JCB file are considered to be increments of current parameter values, they should be centred on one or zero – one for log-transformed parameters and zero for non-log-transformed parameters. For log-transformed parameters, parameter increments are actually multipliers. Hence if increments are generated using the RANDPAR3 utility, the PEST control file on which the running of RANDPAR3 is based should provide initial parameter values that are zero or one as appropriate. Because RANDPAR4 takes care of this automatically, its use is preferable to that of RANDPAR3 when generating random realizations of parameter increments.

It is often convenient to base generation of random parameter increments on the prior parameter probability distribution. However it must be born in mind that prior parameter standard deviations and variances may be too large to use as parameter increments for the purpose of Jacobian matrix computation. PEST_HP allows a uniform factor to be applied to all increments that it reads from a JCB file in order to reduce their values, if a user considers this necessary.

If the value of any adjustable parameter, after adjustment by an externally-supplied increment, exceeds its upper bound, then the value of that parameter is clipped at its bound. However tied parameters are not clipped at their bounds; they continue to respect the same ratio with their parent parameters as that provided in the PEST control file on which the PEST_HP run is based.

9.4.4 Jacobian Matrix Retainment

Optionally, the Jacobian matrix calculated during a given iteration of the PEST_HP inversion process can incorporate part of the Jacobian matrix calculated during the previous iteration of that process. This may reduce errors in this matrix incurred by use of a small number of random parameter increment realizations in its construction. This strategy can also reduce calibration-induced parameter and predictive bias, as it allows the rank of the Jacobian matrix to grow over time as its current contents are based on an ever-increasing number of random parameter increment realizations. However there may be a price to be paid for mixing the current Jacobian matrix with previously-calculated Jacobian matrices. For a highly nonlinear model, and/or for an inverse problem that is profoundly ill-posed, this strategy may compromise PEST_HP's ability to calculate up-to-date Jacobian matrices which support continuous objective function improvement in difficult objective function terrains.

PEST_HP reads a user-supplied control variable named RANDJACRETAIN. This specifies the fraction of the previous Jacobian matrix that is retained in calculating a new Jacobian matrix. Suppose that a rank-deficient Jacobian matrix has just been calculated using equation 9.4 (possibly modified by localization functionality as described above). Let this matrix be denoted as $\mathbf{J}_1$. Let $\mathbf{J}_2$ specify the Jacobian matrix (unimproved by Broyden updating) that was calculated during the previous iteration (which, of course, includes part of the Jacobian matrix calculated during the iteration before that). Then the pre-Broyden-updated Jacobian matrix **J** used in the current iteration is calculated as:

$$\mathbf{J} = \frac{\mathbf{J}_1 + R \times \mathbf{J}_2}{1+R} \tag{9.12}$$

Where *R* is the value of RANDJACRETAIN. As in normal usage of PEST_HP, a modeller can implement Broyden updating of a Jacobian matrix based on model outputs calculated during a single, parallelized round of parameter upgrade testing. Theoretically, deficiencies in this matrix can thereby be partially rectified. A second round of parallelized parameter upgrade testing can then be undertaken using the improved Jacobian matrix. In some circumstances this may result in the calculation of parameter upgrade vectors that effect a larger improvement in the objective function than those calculated using a non-Broyden-updated Jacobian matrix. In other cases it does not. Nevertheless, Broyden updating of the Jacobian matrix is recommended, for although its use incurs a model run cost, experience suggests that it can often get a stalled PEST_HP run moving again.

9.5 JCO and JCR Files

As is usual practice, PEST_HP records a JCO file (i.e. a Jacobian matrix file) at the end of each iteration of the inversion process in which parameters are improved. Hence the JCO file that is present at the end of the inversion process (and at any stage thereof) is that which was used to calculate parameters appearing in the PAR file (i.e. the parameter value file). The filename base of both of these files is that of the PEST control file. The PAR file records best parameters achieved up to any time in the inversion process.

When PEST_HP employs its randomized Jacobian functionality, the composition of the Jacobian matrix that is stored in the JCO file may reflect all of the following processes:

1. The outcomes of equation 9.4 based on the set of random parameter increments used in the most recent iteration of the inversion process;

2. The outcomes of inclusion of a fraction RANDJACRETAIN of the non-Broyden-upgraded Jacobian matrix used in the previous iteration of the inversion process;
3. If undertaken, Broyden improvement of the Jacobian matrix based on a parallelized set of model runs in which upgraded parameters are tested.

The format of a JCR file is identical to that of a JCO file. So, after being renamed with a "*.jco*" extension, it can be read, and its contents manipulated, by PEST-suite software which is designed for this purpose. However there are two major differences between a JCR file written by PEST_HP and a JCO file. They are as follows:

1. The Jacobian matrix recorded in a JCR file is not Broyden-updated;
2. The JCR file is updated during every iteration of the inversion process, regardless of whether or not parameters were improved during that iteration. It comprises the **J** of equation 9.12; in doing so, it includes the effects of automatic and/or user-specified localization if these are employed in the PEST_HP inversion process.

As is reported in PEST documentation, Jacobian matrices have many uses. One of these uses is linear parameter and predictive uncertainty analysis. Whether a rank-deficient Jacobian matrix calculated using the randomized Jacobian functionality of PEST_HP is suitable for use in linear analysis is a matter that is currently under investigation. However if it can be so used, the Jacobian matrix contained in a JCR file may be more suitable for this purpose than that contained in a JCO file. This is because the Broyden upgrading procedure may be very effective in promulgating parameter improvements. However it may distort the Jacobian matrix to some extent.

9.6 Control Variables

9.6.1 Reading the Control Variables

Variables which control the action of PEST_HP's randomized Jacobian functionality are read from a section of the PEST control file that is dedicated to holding these variables. Unsurprisingly, this section is named the "randomized jacobian" section. This section must have at least five lines, the first being a header line. The variables comprising these lines are listed in figure 9.1. An example is provided in figure 9.2.

```
* randomized jacobian
RANDOMJAC  RANDOMSEED
NRANDSETSTART NRANDSETINC NRANDSETFIN PHIREDRANDINC RANDINCFAC
RANDJACRETAIN [AUTOLOC AUTOTHRESH LASTLOCITN LOCJCOFILE]
NUMINCSCHED   RANDSCHEDITN1    RANDSCHEDITN2
The following line must be repeated NUMINCSCHED times.
NUMITN  FRACORIGPHI  INCJCBFILE  RANDINCFAC  RANDJACRETAIN
```

Figure 9.1 Variables appearing in the "randomized jacobian" section of a PEST control file.

```
* randomized jacobian
1    222                  #RANDOMJAC  RANDOMSEED
80  5  100  0.1  2.0       #NRANDSETSTART NRANDSETINC NRANDSETFIN PHIREDRANDINC RANDINCFAC
0.4 2 1.0 5                #RANDJACRETAIN AUTOLOC AUTOTHRESH LASTLOCITN (no LOCJCOFILE)
4     1       999          #NUMINCSCHED   RANDSCHEDITN1    RANDSCHEDITN2
10   0.5    inc1.jcb  0.1  0.05  #NUMITN  FRACORIGPHI  INCJCBFILE  RANDINCFAC RANDJACRETAIN
10   0.1    inc2.jcb  0.1  0.1   #NUMITN  FRACORIGPHI  INCJCBFILE  RANDINCFAC RANDJACRETAIN
100  0.005  fdinc     2.0  0.2   #NUMITN  FRACORIGPHI  INCJCBFILE  RANDINCFAC RANDJACRETAIN
100  0.001  fdinc     2.0  0.3   #NUMITN  FRACORIGPHI  INCJCBFILE  RANDINCFAC RANDJACRETAIN
```

Figure 9.2 An example of the "randomized jacobian" section of a PEST control file.

The "randomized jacobian" section must be placed between the "control data" and "parameter groups" sections of a PEST control file. Other sections (such as "singular value decomposition", "simultaneous parameter increments", "sensitivity reuse" and "rsi") can also appear between the "control data" and "parameter groups" sections of a PEST control file. There is no prescribed ordering for these sections if more than one of them is present.

Note that, at the time of writing, PESTCHEK tolerates the presence of a "randomized Jacobian" section in a PEST control file, but does not check its contents. The PSTCLEAN utility removes this section from a PEST control file in order to render that file inoffensive to PEST and BEOPEST.

The roles of variables appearing in the "randomized Jacobian" section of a PEST control file are now discussed in detail.

9.6.2 RANDOMJAC

This variable (an integer) must be set to either 0 or 1. If it is set to 0, then randomized Jacobian calculation is disabled; hence derivatives are calculated using finite parameter differences in the normal PEST_HP manner. Alternatively, if RANDOMJAC is set to 1, then randomized Jacobian calculation is enabled.

9.6.3 RANDOMSEED

RANDOMSEED is the seed for the random number generator. Select any integer greater than zero.

9.6.4 NRANDSETSTART, NRANDSETINC, NRANDSETFIN, PHIREDRANDINC

The first three of these variables are integers while the last is a real number. They all pertain to the variable *N* (i.e. the number of parameter realizations) featured in equations 9.5, 9.6 and 9.11.

NRANDSETSTART is the initial value of *N*.

If, during a particular iteration of the inversion process, the value of the (measurement) objective function does not improve by more than a relative amount of PHIREDRANDINC of its best value achieved to date, then PEST_HP increases *N* by NRANDSETINC ("INC" stands for "increment"). However *N* will never be allowed to increase above NRANDSETFIN ("FIN" stands for "finish"). A value of 0.1 is often suitable for PHIREDRANDINC.

Some of the matters which need to be considered when selecting values for NRANDSETSTART, NRANDSETINC and NRANDSETFIN are discussed earlier in this chapter. In addition to the theoretical and numerical considerations that were discussed above, a further practical matter requires consideration. PEST_HP commissions model runs on a suite of agent nodes. Sometimes these nodes will have been purchased from a cloud provider. In these inversion circumstances, under-utilization of agent nodes serves no numerical purpose, but increases financial costs. Hence, at any stage of the inversion process, the value of *N* should be an integral multiple of the number of available agents.

9.6.5 RANDINCFAC

If PEST_HP generates realizations of random parameter increments itself, it declares all adjustable parameters to be statistically independent. The mean value of each parameter is its reference value at that particular stage of the inversion process. The standard deviation ascribed to each parameter is equal to the increment that it would have been assigned if derivatives were computed using two-point finite parameter differences. As stated above, these increments are controlled by the DERINC, DERINCLB and INCTYPE control variables that are assigned to parameter groups.

Standard deviations calculated for all parameters (and hence all parameter increments) using the DERINC, DERINCLB and INCTYPE control variables are multiplied by RANDINCFAC prior to the random number generation process. RANDINCFAC is a real number whose value must be greater than zero.

9.6.6 RANDJACRETAIN

This real variable should be set to a value between 0.0 and 1.0. It governs how much of the covariance matrix calculated during the previous iteration of the inversion process is retained for use in calculating parameter upgrades during the current iteration; see equation 9.12. Set RANDJACRETAIN to 0.0 to suppress carryover of the Jacobian matrix between iterations.

Note that, as was stated above, the Jacobian matrix from a previous iteration that is partially retained for use in the current iteration has not undergone Broyden upgrading, even if Broyden upgrading of that matrix was implemented during the previous iteration.

9.6.7 AUTOLOC, AUTOTHRESH and LASTLOCITN

AUTOLOC is an integer variable. If it is set to zero, then autolocalization is disabled. A setting of 1 enables hard autolocalization, while a setting of 2 enables soft autolocalization.

AUTOTHRESH is the value of *A* of equation 9.11 The recommended value of this variable is between 0.1 and 1.0. The lower that AUTOTHRESH is set, the less aggressive is autolocalization; conversely, setting it to a higher value is more aggressive as it results in reduction in the absolute values of a greater number of Jacobian elements (because a greater number of observation-to-parameter correlations fall below the t_α threshold of equation 9.11). The behaviour of PEST_HP in lowering the (measurement) objective function, and an inspection of the autolocalization report file (*case*.*alr*) may suggest that it be raised or lowered from its existing value. Recall that *case.alr* records the number of parameter-to-observation correlation coefficients that are less than the t_α threshold. Corresponding elements of C(**o**,**k**) are set to zero if hard thresholding is enabled, while their values are diminished if soft thresholding is enabled. PEST_HP also records the number of affected Jacobian elements on the screen.

If activated by setting AUTOLOC to 1 or 2, automatic localization is implemented during every iteration of the inversion process using the outcomes of the most recent set of model runs; these are the model runs used for construction of the current C(**o**,**k**) parameter-to-observation covariance matrix. However automatic localization is not implemented for iterations LASTLOCITN+1 and later. Set LASTLOCITN to a large value (e.g. 999) to ensure that autolocalization is implemented throughout the inversion process.

AUTOLOC can be omitted from the "randomized jacobian" section of a PEST control file, so that RANDJACRETAIN is the only variable featured on the fourth line of this section. In this case, AUTOLOC is assumed to be 0. If it is explicitly or implicitly set to zero, then entries for AUTOTHRESH and LASTLOCITN can be omitted. However values for all of these variables must be supplied if a filename is supplied for the LOCJCOFILE variable, for this is (optionally) identified as the fifth item on this line.

9.6.8 LOCJCOFILE

If a user wishes that PEST_HP employ a localization matrix that he/she has prepared him/herself (i.e. the matrix **L** of equation 9.9), then the name of a file containing this matrix must be supplied as the

value of the LOCJCOFILE variable. This must be a JCO file. It may have been written by a program such as JCOBLANK. The matrix contained in this file must have the following specifications:

- It must have dimensions of NOBS by NESPAR, where NOBS is the number of observations (excluding prior information) featured in the current PEST control file, and NESPAR is the number of adjustable parameters featured in the current PEST control file.
- All elements of this matrix must be between zero and one (inclusive).

PEST_HP will object with an error message if either of these conditions is violated.

The localization matrix provided in the LOCJCOFILE file is used for all iterations of the inversion process, regardless of the setting of LASTLOCITN. It can then be used instead of, or as well as, autolocalization.

9.6.9 NUMINCSCHED

NUMINCSCHED stands for "number of increment schedules". An "increment schedule" instructs PEST_HP how to obtain random parameter increments during a particular iteration of the inversion process. As has already been discussed, PEST_HP can calculate increments itself; alternatively it can read them from a user-supplied JCB file. A user can supply a number of different JCB files containing increment realizations with different statistical characteristics. Each of these must be assigned to a different increment schedule.

If NUMINCSCHED is set to zero, then no parameter increment schedules are provided. Hence, throughout the entire inversion process, PEST_HP calculates parameter increments itself based on the assumption of parameter statistical independence using variables supplied in the "parameter groups" section of the PEST control file to express parameter increment stochasticity.

9.6.10 RANDSCHEDITN1 and RANDSCHEDITN2

These are both integer variables They must both be zero or greater. If one of them is zero, then the other must be zero; otherwise RANDSCHEDITN2 must be greater than RANDSCHEDITN1. If RANDSCHEDITN1 and RANDSCHEDITN2 are both greater than zero, then NUMINCSCHED must be greater than 1.

Suppose that RANDSCHEDITN1 is set to i_1 and that RANDSCHEDITN2 is set to i_2. Then during iterations i_1 to i_2 of the inversion process, PEST_HP selects an increment schedule randomly from the first NUMINCSCHED-1 schedules that are provided to it. On iterations following i_2, PEST_HP employs the last parameter increment schedule (i.e. schedule NUMINCSCHED) exclusively. Set RANDSCHEDITN2 to a high number such as 999 if you do not wish that PEST_HP makes exclusive use of this final increment schedule as it approaches the end of the inversion process.

If RANDSCHEDITN1 and RANDSCHEDITN2 are both set to zero, then PEST_HP uses increment schedules in order of their listing in the "randomized Jacobian" section of the PEST control file, changing from one to the next in accordance with the settings of pertinent increment schedule control variables.

9.6.11 INCJCBFILE

INCJCBFILE is a character variable. Optionally, it contains the name of a JCB file which houses an ensemble of parameter increment realizations. (Note that a JCB file must possess an extension of

"*.jcb*".) As has already been stated, this file will normally have been written by the RANDPAR4 utility. Increments should be centred on 0.0 for non-log-transformed parameters and on 1.0 for log-transformed parameters. There should be at least as many increment realizations in this file as PEST_HP needs to use. Hence if, for example, this file will be used over four iterations of the inversion process and the NRANDSETFIN control variable is set to 200, then this JCB file should contain at least 800 realizations of parameter increments. If RANDSCHEDITN1 and RANDSCHEDITN2 are greater than zero, then the number of iterations for which PEST_HP requires increments from this file must be guessed, as use of this file is a random matter. It is better to over-supply increment realizations than to under-supply them; this is not a problem if using RANDPAR4 to fill this file.

Alternatively, INCJCBFILE can be supplied as "fdinc". This instructs PEST_HP to calculate parameter increments itself under the assumption of parameter stochastic independence using variables which would otherwise control the calculation of finite difference derivatives.

9.6.12 RANDINCFAC and RANDJACRETAIN

All increments read from file INCJCBFILE, or their logs, are multiplied by an increment-schedule-specific value of RANDINCFAC before being used by PEST_HP. (Their increments are multiplied if a parameter is untransformed, and their logs are multiplied if a parameter is log-transformed.) Alternatively, if INCJCBFILE is set to "fdinc" so that PEST_HP calculates increments itself, then RANDINCFAC multiplies these increments. This overrides the value supplied for RANDINCFAC on the third line of the "randomized jacobian" section of the PEST control file. The latter variable is employed only if NUMINCSCHED is set to zero.

An increment-schedule-specific fraction RANDJACRETAIN of the Jacobian matrix used in the previous iteration is retained in calculating the Jacobian matrix on which parameter upgrades are calculated during the current iteration. This overrides the value supplied for RANDJACRETAIN on the fourth line of the "randomized jacobian" section of the PEST control file. The latter variable is employed only if NUMINCSCHED is set to zero.

9.6.13 NUMITN and FRACORIGPHI

If RANDSCHEDITN1 and RANDSCHEDITN2 are both set to zero, then PEST_HP uses increment schedules in the order in which they are provided in the "randomized jacobian" section of the PEST control file. It uses a particular increment schedule for NUMITN iterations, or until the (measurement) objective function has fallen to a fraction FRACORIGPHI of its original value. Then it progresses to use of the next increment schedule. Note that, in order for this type of increment scheduling to work, values provided for FRACORIGPHI must reduce with increasing increment schedule number.

9.7 Accommodating Other PEST_HP Functionality

9.7.1 Incompatibilities

Randomized Jacobian functionality cannot be employed under any of the following circumstances:

- PEST_HP is undertaking SVD-assisted parameter estimation, or is run in "pareto" mode;
- The PEST_HP control file instructs PEST_HP to employ different commands for running the model when calculating finite-difference derivatives with respect to different parameters;
- Split slope analysis is used for accommodation of poor numerical derivatives;
- Sensitivity re-use functionality has been enabled;

- Parameter bounds sticking functionality has been enabled.

If you attempt to violate the first two of the above conditions, PEST_HP will cease execution with an error message. In the other three cases, PEST_HP disables the offending functionality itself.

The PEST whisperer (PWHISP_HP) is not yet programmed to read or understand the contents of a PEST run record file recorded by PEST_HP when employing randomized filling of the Jacobian matrix. Nor can the PCOST_HP cost calculator estimate the number of model runs required for parameter estimation based on a randomized Jacobian matrix.

9.7.2 DERFORGIVE

If, during the Jacobian-filling phase of an inversion iteration, the model fails to run to completion while employing a random set of parameter increments, PEST_HP does not cease execution with a pertinent error message. In accommodating model run failure in this way, its behaviour is thus similar to finite-difference filling of the Jacobian matrix with a DERFORIGVE setting of "derforgive". Because the model produces no useable outputs under these circumstances, the covariance matrix of equation 9.5 is not updated in the way that it would be updated if the model run was a success. Because a realization is missing, the accuracy of this matrix is lowered; at the same time, its rank-deficiency is increased, as is the subspace of parameter space in which parameter upgrade vectors may lie.

If, on any iteration of the inversion process, all Jacobian-filling model runs fail, PEST_HP ceases execution with an appropriate error message.

9.8 Experience to Date

9.8.1 Autolocalization

History-matching based on randomized Jacobian calculation appears to benefit from the use of hard or soft autolocalization, with AUTOTHRESH set to between 0.1 and 1.0. This is especially the case if using independent random parameter increments, that is if NUMINCSHED is set to zero. With autolocalization switched on, progress during early stages of an inversion process is often greatly accelerated. However it does not appear to be so beneficial towards the end of the process. Hence a setting of about 5 for LASTLOCITN is often beneficial.

9.8.2 Increment Schedules

Use of a randomly-selected mixture of increment schedules appears to work well. Some of these may be based on alternative concepts of the prior parameter probability distribution, and inter-parameter correlations implied by these. Another increment schedule may rely on independent parameter stochasticity (in which case INCJCBFILE for that schedule may be set to "fdinc"); this may support introduction of locally connected permeability (or locally connected impermeability) to the model domain in ways that are required to fit the calibration dataset, but that were unanticipated by the modeller. If RANDSCHEDITN2 is not set to a large number such as 999, then it may be a good idea to set INCJCBFILE to "fdinit" for the final increment schedule; this allows PEST_HP to spend the last few iterations of the inversion process "filling in the fine detail" of a parameter field in order that model outputs can fit the calibration dataset well.

9.8.3 Solution Method

PEST_HP offers the following methods for calculation of upgraded parameters (see PEST documentation for details):

- Gaussian elimination;
- Singular value decomposition with SVDMODE set to 1;
- Singular value decomposition with SVDMODE set to 2;
- LSQR.

As is discussed elsewhere, use of SVD or LSQR ensures stability of the inversion process, regardless of whether or not the inverse problem is well-posed. As has already been discussed, evidence to date (and some theory), suggests that if NUMINCSHED is set to zero so that increment calculation is based solely on use of finite difference control variables (thus assuming stochastic independence of parameter increments), an SVDMODE setting of 2 works well. However this does not appear to be the case where increment ensembles are based on covariance matrices that are more reflective of possible prior parameter distributions.

9.8.4 Tikhonov Regularisation

Tikhonov regularisation allows expert knowledge to enhance the integrity of the inversion process. Its use can promulgate solution of an inverse problem which approaches that of minimum error variance. This often comprises a “smooth” rather than “bumpy” estimated parameter field. A NUMINCSCHED setting of zero can precipitate the emergence of bumpy parameter fields; however the use of Tikhonov regularization, in conjunction with a covariance matrix applied to pertinent prior information equations, can forestall this occurrence.

As is extensively discussed in PEST documentation, in its implementation of Tikhonov regularisation PEST calculates a factor for regularisation weights which attempts to ensure that the measurement objective function attains, but does not undercut, a user-specified target measurement objective function. This weight factor is re-computed during every iteration of the inversion process. Local linearization of the model (i.e. a Jacobian matrix) is required for calculation of this weight factor. Where the Jacobian matrix is only approximate, the weight factor can only be approximate. Furthermore, its value may undergo significant apparent changes from iteration to iteration, this reflecting random variation of elements of the Jacobian matrix.

Experience to date suggests that PEST_HP performs satisfactorily when using a randomized Jacobian matrix if an inverse problem includes Tikhonov regularisation. However while PEST_HP may be able to achieve an appropriate target measurement objective function, it may not be able to lower the regularisation objective function to as low a value as it could if the Jacobian matrix were filled using finite parameter differences.

Experience has also shown that this problem can be ameliorated to some extent if the PHIMACCEPT regularisation control variable is set a little higher than when using finite-difference derivatives for filling of the Jacobian matrix – perhaps 5 percent to 10 percent higher than PHIMLIM. Once the measurement objective function has fallen below PHIMACCEPT, PEST_HP then concentrates on lowering the regularisation objective function. This can result in a much more acceptable parameter field than would otherwise be the case. In particular, the estimated parameter field may not be as

"bumpy" as it would otherwise be because of null space entrainment - a problem to which use of a randomized Jacobian is particularly susceptible. (See the discussion at the beginning of this chapter.)

An alternative to determination of a regularisation weight factor based on the value of PHIMLIM is to use the REG2MEASRAT regularisation control variable as a basis for calculation of this factor. However, while this is fast, it is also approximate. Nevertheless a simplified calculation of the regularisation weight factor in this manner is less prone to errors incurred by use of a rank-deficient Jacobian matrix than a more complex calculation based on PHIMLIM. In fact, the Jacobian matrix is not used for calculating the regularisation weight factor at all when the latter's calculation is based on REG2MEASRAT, except perhaps in the first iteration of the inversion process; see documentation of this variable elsewhere in this manual.

Yet another alternative which appears to work well in some circumstances is to set the LAMBDA_FOR_REG control variable to 1 in the "regularisation" section of the PEST control file. Under these circumstances, PEST_HP's selection of the regularisation weight factor is experimental rather than prescriptive. See the pertinent section of this manual for further details.

9.8.5 Retainment of the Previous Jacobian Matrix

For both practical and conceptual reasons, RANDJACRETAIN should be set to a value that allows the Jacobian matrix used for calculation of parameter upgrades in any particular iteration to inherit some characteristics of previously-calculated Jacobian matrices. If this is not done, the rank-deficient matrix **J** calculated using equation 9.4 will possess a solution space (orthogonal complement to its null space) that is likely to be somewhat misaligned with the true solution space of the inverse problem, for it can only be comprised of the space that is spanned by the most recent set of random parameter increment vectors. Provided that the number of these parameter increment sets is equal to, or exceeds, the dimensionality of the solution space, solution space misalignment will not necessarily impair the ability of the inversion process to achieve a good fit with a calibration dataset. However considerable entrainment of null space parameter components will be associated with estimation of parameters. Estimated parameters may therefore endure considerable calibration-induced bias, and estimated parameter fields may lose their aesthetic appeal. To the extent that model predictions show null space dependency, they too may exhibit calibration-induced bias.

By accumulating Jacobian matrices calculated using different realizations of parameter increments over multiple iterations, the rank-deficiency of the Jacobian matrix shrinks over time. This allows a PEST_HP inversion process that is based on Jacobian randomization to define a solution space which is more closely aligned with the true solution space of the inverse problem than would be the case if RANDJACRETAIN were set to 0.0.

Based on limited experience to date, RANDJACRETAIN values of between 0.3 and 0.8 appear to work well. A large value for this variable dilutes the ability of the most recent parameter increments to reflect changes in the Jacobian matrix arising from model nonlinearity. A small value may promulgate null-space entrainment for reasons just outlined.

Because retainment of a previous Jacobian matrix has the potential to increase the rank of the Jacobian matrix, the need for autolocalization (if employed) may diminish with iteration count. In fact, if localization is used to set possibly aberrant Jacobian elements to values at or near zero, the noise that is associated with these elements is never given the chance to cancel through mixing with

elements of previously-calculated Jacobian matrices. This strengthens the case for setting LASTLOCITN to a value of 4 or 5, so that the effects of autolocalization can diminish as the benefits of Jacobian retainment grow.

9.8.6 Broyden Jacobian Updating

Experience to date suggests that in some inversion contexts Broyden Jacobian updating can enhance PEST_HP's performance dramatically when it employs a randomized Jacobian matrix. However there are other cases where it does not.

If Broyden Jacobian updating is not employed, this can sometimes result in slower, but steadier, progress of the inversion process, particularly when employing Tikhonov regularisation. In the latter case, regularisation objective functions appear to be lower for a given measurement objective function than if Broyden updating of the Jacobian matrix is employed. This, of course, is desirable, as Tikhonov regularisation poses the inverse problem as a constrained minimization problem, in which the regularisation objective function is minimized subject to the measurement objective function attaining its user-specified target value.

Nevertheless, as stated above, the benefits of Broyden Jacobian updating in achieving a low measurement objective function often outweigh its costs. These costs are the increased number of model runs that are required for solution of the inverse problem, and a possibly higher regularisation objective function. In areas of high expected parameter spatial heterogeneity, the ability to reflect the presence of this heterogeneity through achieving a low measurement objective function may far outweigh the benefits of parameter field smoothness.

9.8.7 Parameter Change Limits

Limited experience to date suggests that an inversion process based on a randomized Jacobian matrix may benefit from the imposition of tighter parameter change limits than would otherwise be the case. This prevents the occurrence of large, but erratic, changes in parameter values, particularly during early iterations of an inversion process. Consider employing RELPARMAX and FACPARMAX settings of between 3.0 and 5.0. Unfortunately, this measure may increase the number of model runs required for solution of an inverse problem. Fortunately, however, limited experience to date also suggests that the need for this measure is diminished if autolocalization is employed.