523970

May 25th, 2002

QA: QA

# DESIGN DOCUMENT (DD)

for

**PEST Version 5.5**

SAN:
STN:
SMN:
SDN:

Prepared by:______________________________Date________

John Doherty
Watermark Numerical Computing
for
Geoanalysis Group, EES-5
Los Alamos National Laboratory
Los Alamos, New Mexico

Reviewed by: ______________________________Date________

Zora Dash
Technical Reviewer

Approved by: ______________________________Date________

George Zyvoloski
Responsible Manager

Reviewed by: ______________________________Date________

IT Software Management Analyst

**Los Alamos National Laboratory**

# Table of Contents

# 1. Purpose

The purpose of this Design Document (DD) is to comply with the requirements set out in AP-SI.1Q Rev. 3, ICN 3 (*Software Management*) for development and qualification of software. The design outlined herein is intended to comply with the requirements set out in the PEST v5.5 Software Activity Plan and the PEST v5.5 Requirements Document (RD).

# 2. Implementation Standards and Environment Specifications

## 2.1 Programming Standards

PEST and its utilities will be written in FORTRAN. The FORTRAN 77 standard will be observed as closely as possible, with the following exceptions:-

- some variable names will be greater than 6 characters in length;
- some of the code will be written in lower case;
- a non-standard function call will be used to run a model from within PEST;
- a non-standard function call will be used to ascertain command-line arguments;
- non-standard function calls will be used to ascertain the date and time as part of the methodology required to undertake parallel model runs; and
- data arrays will be dynamically allocated.

Where the FORTRAN 77 convention is not observed, the FORTRAN 90 convention will be adhered to, except for the code required to run the model from inside of PEST and the code required to access command line arguments; a FORTRAN 90 standard does not exist for either of these. Fortunately, as most compilers use an almost identical protocol for these functions, a de facto standard exists.

It will be ensured that any non-standard code within PEST will conform to the requirements of the FORTRAN 77 and FORTRAN 90 compilers presently used by the Geoanalysis Group.

## 2.2 System Software

Because of its adherence to FORTRAN coding conventions, executable versions of PEST and its utilities will be easily generated for any operating system for which a FORTRAN compiler is available. This includes the SunOS UNIX operating system employed on machines presently used by personnel of the Geoanalysis Group. However executable files for the PC version of PEST and its utilities will be provided for use with the WINDOWS 2000 operating system. These executable files will be 32 bit WINDOWS executables. They will be capable of being run either from the command line, or through clicking on pertinent file icons from within WINDOWS explorer. If desired, access to these executables can also be made available through the WINDOWS-2000 "Start" menu by following instructions provided in the WINDOWS 2000 help system.

PEST will not be required to interact with any external software systems such as database managers. All user interaction will take place through ASCII input and output files, which can be edited and displayed by the user with standard text editing software.

## 2.3 Hardware

PEST and its utilities will be capable of running on any hardware and operating system for which there are FORTRAN compilation facilities. In particular, they will run on any of the Sun workstations presently operated by the Geoanalysis Group. A makefile will be supplied with PEST to expedite the generation of executable files for this environment.

Executable programs supplied with the PC version of PEST will run on any computer on which the WINDOWS 2000 operating system is installed.

## 2.4 Installation and Validation Methodology

Installation will consist of reading the main program and all required supporting routines and auxiliary files into disk storage on the target platform. Validation will be achieved by executing pre-defined problems for which valid results are available. The installation test consists of execution of a single test problem to demonstrate that the code is executing correctly on the target platform. The validation methodology consists of execution and verification of results from a series of tests that exercise the mathematics of the code and all of its model and user interface functionality. Details of the installation and validation methodologies are given in the "Software Installation Test Plan for PEST Version 5.5" (?????-ITP-5.5-00) and the "Validation Test Plan for PEST Version 5.5" (?????-VTP-5.5-00).

# 3. Software Structure

## 3.1 Functional Requirements

The functional requirements satisfied by the PEST program are summarized in Table 1. The section numbers in column 2 of this table refer to the requirements document (?????-RD-5.5-00). The components of the program (software modules), which implement these requirements, are listed in the third column of the table.

| **Requirement** | **Section** | **Implemented by software modules** |
|---|---|---|
| Inversion Algorithm | 1.2 | *bnderr derclc daxpy daxpy dpofa dposl drvrd dscal gpread main obgprd objclc obsrd oread pgetcl prmrd prrclc prrrd ppstop pstop,rotate, trntyp* |
| Communication between PEST and an existing simulation model | 1.3 | *cmprss extjac getint getnum getnxt gettot inwrit ioctl linspl lowcas model numrd outrd parnam pestmess remchar shiftl spacesub tabrem tabrep upcas which1 wrtsig zinctest* |
| Screen and file output | 1.4 | *ffopen prmsav prmwrt psiwrt psterr stopress stperr wrtall wrtfin wrtmat* |
| User intervention | 1.5 | *hldread main ppause pstop pstopst punpause* |
| Statistical calculations | 1.6 | *main tred2 tql2 wrtfin* |
| Predictive analysis | 1.7 | *main* |
| Regularization | 1.8 | *main optwt* |
| Parallelization | 1.9 | *closefile delfile doruns getsecs pinwrit poutrd pslave slavdat1 slavdat2 slavdat3 sstop wait* |
| Utility Programs | 1.10 | *allsam illins illprr illsgn inschek jacwrit lohi mkrtio par2par parchk parrep pestchek pestgen prmchk prrchk rderr tempchek wrterr wrtrl zroneg zroone* |

**Table 1. Functional Requirements of the PEST program.**

## 3.2 Software Modules

Table 2 lists the modules used by PEST, together with a brief description of the role of each.

| **Module name** | **Role** |
|---|---|
| *allsam* | Checks whether two prior information equations contain the same information. |
| *bnderr* | Checks that array dimensions do not exceed maximum dimensional bounds. |
| *closefile* | Closes a file and checks that it is properly closed. |
| *cmpress* | Compresses instruction set to minimum memory for efficient internal storage. |
| *daxpy* | Calculates constant times a vector plus a vector. |
| *delfile* | Deletes a file and checks that it is properly deleted. |
| *derclc* | Carries out finite-difference derivatives calculation. |
| *doruns* | Organizes carrying out of parallel runs. |
| *dpodi* | Calculates determinant and inverse of positive definite matrix. |
| *dpofa* | Factors a positive definite matrix. |
| *dposl* | Solves Ax=b where A is positive definite. |
| *drvrd* | Reads information from PEST control file pertaining to derivatives calculation. |
| *dscal* | Computes determinant of a positive definite matrix. |
| *extjac* | Reads the "derivatives file" supplied by a model if this is available. |
| *ffopen* | Opens a file. |
| *getint* | Retrieves next instruction line from internal storage. |
| *getnum* | Retrieves the numerical part of an instruction. |
| *getnxt* | Retrieves the next instruction on a specific instruction line. |
| *getsecs* | Obtains elapsed seconds for run. |
| *gettot* | Determines exact position occupied by a number in an instruction. |
| *gpread* | Reads a line of derivative data from PEST control file. |
| *hldread* | Reads and processes data in parameter hold file. |
| *illins* | Reports an error in an instruction. |
| *illprr* | Report an error in prior information. |
| *illsgn* | Checks for errors in the sign of prior information coefficients. |

| | |
|---|---|
| *inschek* | Checks a pest instruction file for errors or inconsistencies. |
| *inwrit* | Writes model input files. |
| *ioctl* | Prepares for reading an instruction set. |
| *jacwrit* | Reads a binary Jacobian matrix file written by PEST and re-writes the same information in ASCII format. |
| *linspl* | Splits a line into space-delimited fragments. |
| *lohi* | Reports whether a value is too low or too high. |
| *lowcas* | Converts a character string to lower case. |
| *main* | The PEST main program; carries out much of the numerical work involved in parameter estimation and predictive analysis. |
| *mkrtio* | Assists in checking the integrity of prior information. |
| *model* | Runs the model. |
| *numrd* | Reads a number from part of a character string. |
| *obgprd* | Reads information pertaining to observation groups from PEST control file. |
| *objclc* | Calculates current value of objective function. |
| *obsrd* | Reads information pertaining to observations from PEST control file. |
| *optwt* | Solves for optimum regularization weight factor. |
| *oread* | Reads a line of observation data from PEST control file. |
| *outrd* | Reads model output files after a model run. |
| *par2par* | Computes a "secondary" set of parameters from a "primary" parameter set based on arbitrary mathematical relationships between the two parameter sets. |
| *parchk* | Checks parameter spaces on template files. |
| *parnam* | Extracts a parameter name from a string. |
| *parrep* | Writes a new PEST control file using an old one, together with a parameter value file. |
| *pestchek* | Checks the integrity of an entire PEST input data set. |
| *pestmess* | Writes a PEST-to-model message file. |
| *pestgen* | Builds a PEST control file based on a parameter value file and an observation value file. |
| *pgetcl* | Reads PEST command line. |
| *pinwrit* | Writes model input files to slave working directories. |

| | |
|---|---|
| *poutrd* | Reads model output files from slave working directories. |
| *ppause* | Allows the user to pauses PEST execution. |
| *pread* | Reads a line of parameter data from PEST control file. |
| *prmchk* | Checks the parameter data section of the PEST control file. |
| *prmrd* | Reads information pertaining to parameters from PEST control file. |
| *prmsav* | Saves optimized parameter values. |
| *prmwrt* | Writes current parameter value to a space. |
| *prrchk* | Checks all prior information in a PEST control file. |
| *prrclc* | Calculates current value of prior information equations. |
| *prrrd* | Reads information pertaining to prior information from PEST control file. |
| *psiwrt* | Records current value of objective function. |
| *pslave* | PEST slave program. |
| *psterr* | Writes a PEST error message. |
| *ppstop* | Handles premature cessation of execution on encountering an error condition. |
| *pstopst* | Allows the user to terminate PEST execution with a statistical printout. |
| *punpause* | Allows the user to re-commence PEST execution after a pause. |
| *rderr* | Writes an error message pertaining to an inability to read a number from a model output file. |
| *remchar* | Removes a specified character from a string. |
| *rotate* | Rotates a covariance matrix to align it with the directions of its principal components. |
| *shiftl* | Left-justifies a string. |
| *slavdat1* | Reads initial part of run management file. |
| *slavdat2* | Reads second part of run management file; looks for slaves. |
| *slavdat3* | Reads optional third part of run management file. |
| *spacesub* | Substitutes a specific character for spaces in a string. |
| *sstop* | Prints message to screen when PSLAVE stops running. |
| *stopress* | Detects message from user to pause or resume execution. |
| *stperr* | Prepares formatting of a PEST error message. |

| | |
|---|---|
| *tabrem* | Removes tabs from a string. |
| *tabrep* | Replaces tabs in a string by spaces while maintaining formatting. |
| *tempchek* | Checks the integrity of a template file. |
| *tred2* | Reduces real symmetric matrix to tridiagonal form. |
| *tql2* | Computes eigenvalues and eigenvectors of matrix. |
| *trntyp* | Ascertains transformation type of each parameter. |
| *upcas* | Converts a string to upper case. |
| *wait* | Waits a user-specified time before further processing. |
| *which1* | Determines the index of a parameter or observation from its name. |
| *wrtall* | Records all input information on run record file. |
| *wrterr* | Formats and writes a PESTCHEK error message. |
| *wrtfin* | Calculates statistics and completes writing of run record file at end of PEST run. |
| *wrtmat* | Writes the matrix file comprised of the covariance matrix, correlation coefficient matrix, and eigenvalues/eigenvectors of the covariance matrix. |
| *wrtrl* | Writes a real number to a character string. |
| *wrtsig* | Writes a number into a restricted space with maximum precision. |
| *zinctest* | Ensures that an incremented parameter has a different value to unincremented parameter when written to model input file. |
| *zroneg* | Writes an error message pertaining to a number which should not be zero or negative. |
| *zroone* | Writes an error message pertaining to a number which should be between zero and one. |

**Table 2. Subroutines used by PEST and its utilities.**

## 3.2 Optimization Algorithm

### 3.2.1 Modes of Operation

PEST will operate with existing models, communicating with these models through their own input and output files, and running them through system calls whenever it needs to know the values of certain model outcomes based on a current set of parameter values.

PEST will operate in three different modes, these being:-

- parameter estimation mode,
- predictive analysis mode, and

- regularization mode.

The operation of these modes is now described.

### 3.2.2 Parameter Estimation Mode

Used in this mode, PEST will minimize the sum of weighted squared differences between model outputs and corresponding field or laboratory measurements using the Gauss-Marquardt-Levenberg (GML) method, as documented in texts such as Bard (1974), Mikhail (1976), Nash and Walker-Smith (1987) and Koch (1988). The sum of weighted squared residuals is referred to as the "objective function".

### 3.2.3 Predictive Analysis Mode

When operated in this mode, PEST will maximize or minimize a key model prediction while simultaneously ensuring that the discrepancy between model outputs and corresponding field measurements (i.e. the objective function) when the model is run under historical conditions remains below a user-specified threshold. The methodology will be based on that presented in Cooley and Vecchia (1987) and Vecchia and Cooley (1987).

### 3.2.4 Regularization Mode

When operated in this mode, PEST will minimize a "regularization objective function" (normally calculated as the sum of weighted squared differences between certain simple functions of spatially-dependent parameter values and corresponding idealized values for these functions based on geostatistical, smoothness or other presumptions), at the same time as it ensures that a "measurement objective function" (the sum of weighted squared differences between model outputs and corresponding field measurements) remains below a user-specified threshold. The theory underlying the algorithm implemented in PEST will be similar to that outlined in de Groot-Hedlin and Constable (1990).

## 3.3 Calculation of Derivatives

### 3.3.1 General

Though achieving different aims, the algorithmic bases of all of PEST's modes of operation will share certain mathematical similarities. Implementation of all of these modes will require that parameter values be iteratively improved on the basis of a set of successive linearity assumptions from initial parameter values supplied by the user. Linearization of the parameter estimation, predictive analysis and regularization problems will be achieved through representing the action of the model by a "Jacobian Matrix", i.e. a matrix whose elements are comprised of the derivatives of every model output for which there is a complementary observation with respect to every parameter whose value can be adjusted through the optimization process.

PEST will obtain parameter derivatives in either of two ways; both of these ways of obtaining derivatives will be possible within the one parameter estimation process undertaken by PEST.

The first method of obtaining parameter derivatives will require that PEST calculate derivatives of model outputs with respect to adjustable parameters by varying each such parameter in turn and undertaking a model run on the basis of the incrementally-varied parameter. Derivatives of model outputs with respect to each varied parameter will then be calculated by finite differences in one of four possible ways. Selection of the appropriate methodology for a particular case will be at the discretion of the user; however, as is discussed below, PEST will have provision to switch from a less accurate to a more accurate methodology when it detects the need for greater precision in derivatives calculation during the course of the optimization process.

The second method of obtaining parameter derivatives available to PEST will be for these derivatives to be calculated internally by the model and provided to PEST in an appropriately formatted ASCII file. Where this is possible it will have the following advantages:-

1. Model-calculated derivatives are often more accurate than those calculated by the method of finite differences; in some parameter estimation contexts, this will increase the efficiency of the inversion process undertaken by PEST.

2. A model can often calculate derivatives internally faster than PEST can calculate them externally through finite differences. In these circumstances, use of PEST's "external derivatives functionality", will result in greater overall PEST execution speed.

The use of both of these methods of derivatives calculation will now be discussed in greater detail. In the following paragraphs the term "current parameter values" will refer to the parameter values being used by PEST at the current stage of the iterative process by which optimized parameter values are calculated from initial parameter values supplied by the user.

### 3.3.2 Finite-Difference-Calculated Derivatives

#### *3.3.2.1 Forward Differences*

Using this method of derivatives calculation, each parameter will be varied upwards from its current value by an increment calculated by PEST on the basis of a user-supplied set of variables which govern derivatives calculation. For a particular model outcome for which there is a corresponding field measurement, the derivative of that output with respect to the incrementally-varied parameter will be calculated as the difference in model outputs calculated on the basis of the incremented and current parameter value, divided by the difference in parameter values (i.e. by the parameter increment).

#### *3.3.2.2 Parabolic Method*

Where necessary, derivatives with respect to a certain parameter will be calculated on the basis of three model runs instead of two in order to achieve greater precision. Two of these runs will be undertaken on the basis of parameter values which are slightly different from the current parameter value; normally, one of these runs will be undertaken with the parameter incremented, while the other will be undertaken with the parameter decremented from its current value. The third model run (common to all parameters) will be that undertaken on the basis of current parameter values. Parabolic interpolation will then be undertaken between the three sets of model outputs generated on the basis of the three parameter values. The derivative at the current parameter value is then calculated on the basis of a parabolic interpolation between these model outputs.

### *3.3.2.3 "Outside Points" Method*

This method, too, will rely on the existence of model outputs calculated for a parameter which is incremented, and then decremented, from its current value. However in this case only two points will be used in calculating the derivative of each model output with respect to that parameter, viz. the model outputs corresponding to the incremented and decremented parameter values. (It can be shown that, even though only two points are used rather than three in calculating the derivative, the fact that the parameter values used in this calculation subtend the current value causes a more accurate approximation to the derivate to be obtained than that calculated on the basis of the forward difference where the current parameter value is only incremented but not decremented.)

### *3.3.2.4 "Best Fit Method"*

This method will use three parameter values - the current parameter value together with an incremented and then a decremented parameter value. Implementation of the "best fit" method of derivatives calculation will require that PEST calculate a line of best fit between the corresponding model outputs. The slope of this line will approximate the derivative.

### *3.3.2.5 Selection of Parameter Increments*

If the parameter increment used to calculate derivatives for any of the above four methods is too large, the outcome of that calculation will be a poor mathematical approximation to the derivative. If the increment is too small, numerical precision will be lost through the differencing of quantities of similar magnitude. Hence selection of the size of the derivative increment will be a matter of some importance when using PEST.

PEST will allow the user to supply values for a number of input variables (viz. DERINC, DERINCLB, DERINCMUL and INCTYP) which will govern the way in which parameter increments are calculated at any stage of the parameter estimation process. Options will include the following:-

- calculation of the increment as a proportion of the current parameter value (i.e. a "relative increment"),
- use of an increment that is independent of the current parameter value (i.e. an "absolute increment"),
- use of a relative increment with an absolute lower bound, and
- calculation of the increment relative to the parameter of highest magnitude within the user-defined group to which the parameter belongs.

### *3.3.2.6 Switching from Two-Point to Three-Point Derivatives Calculation*

PEST will allow the user to determine, on a parameter-by-parameter basis, whether derivatives are to be computed using two-point or three-point derivatives calculation, or whether the optimization process should begin with two-point derivatives calculation and then switch to three-point derivatives calculation for a particular parameter when progress of the optimization process begins to falter. Assessment of the latter condition will depend on PEST's mode of operation. When operating in parameter estimation and regularization modes, PEST will switch to the use of three-point derivatives calculation for those parameters that the user has designated as "switchable" if the objective function has fallen by less than a user supplied relative amount (PEST input variable PHIREDSWH) between successive optimization iterations. When operating in predictive analysis mode, the switch to three-point derivatives calculation will be made if the maximum or minimum prediction being sought by PEST is raised or lowered on successive optimization iterations by less than a user-designated relative or absolute amount (PEST input variables RELPREDSWH and ABSPREDSWH respectively).

### *3.3.2.7 Multiple Model Command Lines*

In some instances (especially where the model run by PEST is actually comprised of a batch or script file), it may be efficient for the execution path taken by the model to differ slightly depending on whether the model is being currently run for the purpose of calculating outputs on the basis of an upgraded parameter set, or to calculate incrementally-varied outputs on the basis of the incrementally varied value of one parameter. In the latter case, gains in efficiency may be possible if those parts of the model that do not depend on the incrementally varied parameter are not executed, thereby saving the CPU time required to undertake these wasted calculations. This may be achievable through running the model using different commands for different purposes; the command used for a particular model run depending on the parameter whose derivative is currently being calculated.

If the PEST variable NUMCOM is set to a value greater than 1, then NUMCOM model commands will need to be listed in the "model command line" section of the PEST control file. A value for the DERCOM variable will then need to be supplied for each parameter in the "parameter data" section of the PEST control file. This is the command number (with commands counted in order of appearance in the "model command line" section of the PEST control file), which PEST will use to run the model with the pertinent parameter incrementally varied in order to calculate derivatives with respect to that parameter. Where three-point derivatives calculation is being undertaken, the model will be run twice in succession using this same command.

### 3.3.3 External Derivatives

Where a model can calculate some or all of the derivatives of its key outputs with respect to its parameters, it will often be better for PEST to use these model-calculated derivatives in implementation of the GML method, than derivatives which it calculates itself using finite parameter differences. If a model can be programmed to calculate these derivatives, and then to write them to an "external derivatives file", PEST will read the latter file to obtain these derivatives. An example of an external derivatives file is shown in Figure 1.

```
4  9
5.00000  1707.60  34.4932  42.1234
5.25066  8.79458  93.2321  23.5921
1.04819  1.16448  5.34642  19.3235
1.52323  0.11418  0.59235  75.2354
3.21342  0.48392  9.49293  95.3459
2.49321  5.39230  0.49332  9.22934
19.4492  9.93024  0.49304  5.39234
36.3444  10.4933  0.59439  6.49345
95.4592  86.4234  47.4232  324.434
```

**Figure 1. An external derivatives file.**

The first line of an external derivatives file will contain two integers listing the number of parameters and number of observations represented in the file. These must correspond to the number of columns and the number of rows respectively in the derivatives matrix. They must also agree exactly with the values of the PEST variables NPAR and NOBS cited in the PEST control file, i.e. the number of parameters and number of observations respectively involved in the parameter estimation process. The derivative matrix will be listed next in the file.

External derivatives calculation functionality will be activated through the provision of a non-zero value for the variable JACFILE which will reside in the "control data" section of the PEST control file. In this case the PEST control file must contain a "derivatives command line" section comprised of two data lines. The first of these two lines will be the command which PEST will use to run the model for the purpose of derivatives calculation. The second line will contain the name of the file to which the model will record the derivatives which it calculates, that is, the external derivatives file. (This will be the PEST variable EXTDERFLE.)

### 3.3.4 PEST-to-Model Messaging

As is described elsewhere in this document, PEST's principal means of communication with a model will be through the model's own input and output files. However sometimes it will be possible to achieve gains in efficiency through more sophisticated communication between PEST and the model. Such gains will be principally achieved through the optimization of derivatives calculation either through having these directly calculated by the model itself, or through allowing the model to adjust its behavior in accordance with the parameter whose derivative is being currently calculated by PEST using finite differences.

As has already been discussed, PEST will have the capacity to run the model using different commands in order that the model can vary its behavior in accordance with the purpose of its current run. However PEST will provide an alternative means of communication with a model, this being through a "message file" which will provide the model with enough information for it to adjust its behavior, if necessary, in accordance with PEST's current processing requirements.

Figure 2 shows an example of a PEST-to-model message file.

```
derivative_increment
     -2
      4      20
     hcond1       5.005787            1
     hcond2       9.850230            0
     stor1       -5.660591           -2
     stor2        8.257257       -10000
```

**Figure 2. A PEST-to-model message file, *pest.mmf*.**

The first line of a message file will contain a character string which provides information on the purpose of the current model run. The various strings which will be used by PEST are as follows:-

*forward_model_run*

This string will inform the model that it is being run either to test a parameter upgrade, or as the first model run of the optimization process.

*derivative_increment*

This will inform the model that it is being run as part of the finite-difference derivatives calculation process undertaken by PEST.

*external_derivatives*

The model is being run in order to write an external derivatives file.

If the character string on the first line of the PEST-to-model message file is "derivative_increment", then the integer on the second line of this file will be significant. A value of *n* for this integer will indicate that the model run is being undertaken with the value of the $n^{th}$ parameter incremented for the purpose of calculating derivatives with respect to that parameter by forward differences, or as the first of two runs by which derivatives will be calculated using central differences. A value of *-n* will indicate that the $n^{th}$ parameter is currently decremented in the second of two runs undertaken for the purpose of derivatives calculation by central differences.

The third line of the message file will list the number of parameters (PEST variable NPAR) and number of observations (PEST variable NOBS) involved in the parameter estimation process. Following this will be NPAR lines of data with three entries on each line. The first entry on each line will be a parameter name (up to 12 characters in length). Then will follow the value of that parameter used for the current model run. Following that will be an integer code that informs the model of the parameter's status in the inversion process. A value of 0 will denote that the parameter is adjustable and is not logarithmically transformed (see Section 3.12). A value of 1 will indicate that the parameter is adjustable and is logarithmically transformed. A value of *-n* will indicate that the parameter is tied to parameter number *n*, while a value of -10000 will indicate that the parameter is fixed.

The PEST-to-model message file will always be named *pest.mmf* and will be written to the current working directory; it is written just before each model run is undertaken. However in the case of Parallel PEST (see Section 3.17), the message file will be written to each slave working directory just before the pertinent model run is initiated by the slave.

## 3.4 Upgrading the Parameter Set

### 3.4.1 Upgrade Formulae

When working in parameter estimation mode, PEST will calculate a "parameter upgrade vector" using the formula:-

$$\mathbf{u} = (\mathbf{J}^t\mathbf{Q}\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^t\mathbf{Q}\mathbf{r} \qquad (1)$$

where:-

| | |
|---|---|
| **u** | is a vector whose components are the numbers to be added to current parameter values in order to calculate updated parameter values (which, hopefully, will result in a lower objective function), |
| **J** | is the Jacobian matrix, |
| **Q** | is the "cofactor" matrix, a diagonal matrix whose elements are the inverse of weights applied to measurements comprising the calibration data set, |
| **r** | is a vector containing the current values of residuals (i.e. the differences between model outputs and corresponding field measurements), |
| **I** | is the identity matrix, and |
| $\lambda$ | is the "Marquardt lambda"; see the next section for a further discussion. |

Note that in this and all other equations presented in this document, the "t" superscript indicates the transpose of a matrix.

When working in regularization mode, PEST will use the same formula for updating parameter values. However, during each optimization iteration it will adjust the elements of the cofactor matrix **Q** in order that the weights applied to the "regularization observations" are such as to guarantee that the "measurement objective function" rises no higher than either the user-specified upper limit for this quantity (PEST input variable PHIMLIM), or a factor (given by the PEST variable FRACPHIM) of the current value of the objective function, whichever is higher.

When working in predictive analysis mode, PEST will update parameter values using the formula:-

$$\mathbf{u} = (\mathbf{J}^t\mathbf{Q}\mathbf{J} + \lambda\mathbf{I})^{-1}\,[\mathbf{J}^t\mathbf{Q}\mathbf{r} - \mathbf{Z}/2\mathrm{a}] \qquad (2)$$

where *a* is calculated using the equation:-

$$\left(\frac{1}{2a}\right)^2 = \frac{\Phi_0 - \mathbf{r}^t\mathbf{Q}\mathbf{r} + \mathbf{r}^t\mathbf{Q}\mathbf{J}\left(\mathbf{J}^t\mathbf{Q}\mathbf{J}\right)^{-1}\mathbf{J}^t\mathbf{Q}\mathbf{r}}{\mathbf{Z}^t\left(\mathbf{J}^t\mathbf{Q}\mathbf{J}\right)^{-1}\mathbf{Z}} \tag{3}$$

Variables used in equations (2) and (3), additional to those used in equation (1), are as follows:-

| | |
|---|---|
| **Z** | represents the linearized action of the model acting under predictive conditions, |
| *a* | is a variable calculated as part of the prediction maximization/minimization process, and |
| $\Phi_0$ | is the user-supplied upper objective function limit (PEST input variable PD0). |

To achieve maximum accuracy, PEST will optionally perform a line search along the direction of the vector defined by **u** in equation (2), to maximize or minimize the key model prediction while respecting the $\Phi_0$ constraint. The PEST input variables INITSCHFAC, MULSCHFAC and NSEARCH will govern PEST's implementation of this line search.

### 3.4.2 The Marquardt Lambda

Both of equations (1) and (2) involve the use of the Marquardt lambda. The optimum value of this variable to be used at any stage of the optimization process will be calculated by PEST. During each optimization iteration, after it has formulated the Jacobian matrix **J**, PEST will calculate parameter upgrade vectors using either of equations (1) or (2) (depending on its mode of operation) for a number of different values of the Marquardt lambda; by doing this it will be able to determine the best value to use for this variable at the current stage of the optimization process. Variables governing the process by which PEST selects new Marquardt lambdas for testing in this manner, and the criteria by which it decides that enough Marquardt lambdas have been tested, will be supplied by the user with the PEST input data set (PEST input variables RLAMBDA1, LAMFAC, PHIRATSUF, PHIREDLAM and NUMLAM). During any one optimization iteration PEST will continue to test the effects of new Marquardt lambdas until one of the following conditions are met:-

- the maximum number of Marquardt lambdas (NUMLAM) has been tested,
- the relative improvement in the objective function between successive Marquardt lambdas is less than PHIREDLAM, or
- an objective function is calculated which is less than a fraction PHIRATSUF of the lowest objective function achieved during the previous optimization iteration.

## 3.5 Selection of Regularization Weights

When working in regularization mode, PEST will have an additional task to perform during each optimization iteration in that it must calculate a "regularization weight factor"; this is the factor by which the weights associated with all "regularization observations" will be multiplied in order to guarantee that the "measurement objective function" (i.e. the sum of squared weighted differences between model outputs and actual field measurements) is raised no higher than the threshold PHIMLIM.

The regularization weight factor will be calculated using an iterative procedure based on linearization of the problem at the current parameter values. PEST input variables which will govern the operation of this iterative solution process will be named WFFAC (factor by which the weight factor is initially adjusted during this process) and WFTOL (the iterative solution convergence criterion).

## 3.6 Termination Criteria

The outcome of every optimization iteration undertaken by PEST will be an upgraded set of parameter values which, hopefully, will result in a lower value of the objective function than that calculated during the previous optimization iteration (if PEST is working in parameter estimation mode) or a higher/lower model prediction within the constraint set by $\Phi_0$ of equation (2) (if PEST is working in predictive analysis mode). Once PEST has decided that no more Marquardt lambdas will be tested, it must then decide whether to proceed to another optimization iteration, or to declare the optimization process as complete.

Termination criteria will vary slightly between modes of operation. When working in parameter estimation or regularization modes, termination criteria will be determined by PEST input variables NOPTMAX, PHIREDSTP, NPHISTP, NPHINORED, RELPARSTP and NRELPAR. PEST will terminate execution if any of the following criteria are met:-

- over the last NPHISTP optimization iterations, the objective function has been lowered by a relative amount of no more than PHIREDSTP,
- NPHINORED optimization iterations have elapsed since the objective function was last lowered,
- the value of no parameter has been altered by a relative amount of more than RELPARSTP over the last NRELPAR optimization iterations, or
- NOPTMAX optimization iterations have been carried out.

When working in predictive analysis mode, termination criteria will be determined by the input variables NPREDNORED, ABSPREDSTP, RELPREDSTP and NPREDSTP. If the value of the key model outcome which PEST is attempting to maximize or minimize within the $\Phi_0$ constraint changes by no more than an absolute amount of ABSPREDSTP or a relative amount of RELPREDSTP over NPREDSTP optimization iterations, or has not been raised/lowered over the last NPREDNORED iterations, PEST will terminate the optimization process.

## 3.7 Structure of an Optimization Algorithm

Figure 3 illustrates the steps to be taken by PEST during each optimization iteration. PEST's first task during an optimization iteration will be to compute the Jacobian matrix. To do this, PEST will need to run the model with which it is working at least as many times as there are adjustable parameters. Where three-point derivatives calculation is employed, two model runs for each adjustable parameter will be required. Alternatively, if PEST's external derivatives functionality is invoked, PEST will obtain model-calculated derivatives by reading the external derivatives file generated by the model.

If PEST is working in regularization mode, it will then calculate the optimum regularization weight factor to employ for that particular optimization iteration. Once this has been calculated PEST will then calculate a parameter upgrade vector on the basis of the current Marquardt lambda, and will run the model in order to calculate an objective function on the basis of the upgraded parameter set (as well as the value of the key model prediction if it is working in predictive analysis mode). It will then lower the Marquardt Lambda, calculate a new parameter upgrade vector and undertake another model run in order to calculate the objective function corresponding to this new parameter set (and the key model prediction if working in predictive analysis mode). PEST will then continue to test the effects of new Marquardt lambdas (lowering it if possible, but raising it if necessary) until one of the conditions discussed above for terminating the testing of further Marquardt lambdas has been achieved.

If, for a particular optimization iteration, there are any parameters for which derivatives are currently being calculated using forward differences, but for which the user has indicated that one of the three-point methods is to be used if there is an insufficient improvement in the objective function, PEST will then test whether the objective function improvement has been poor enough to justify making the switch to three-point derivatives calculation. If so, it will set a flag so that all future Jacobian calculations will employ three-point derivatives calculation for such parameters, and will then proceed to the next optimization iteration. Otherwise it will check whether any termination criteria have been met. If not, it will then begin the next optimization iteration.

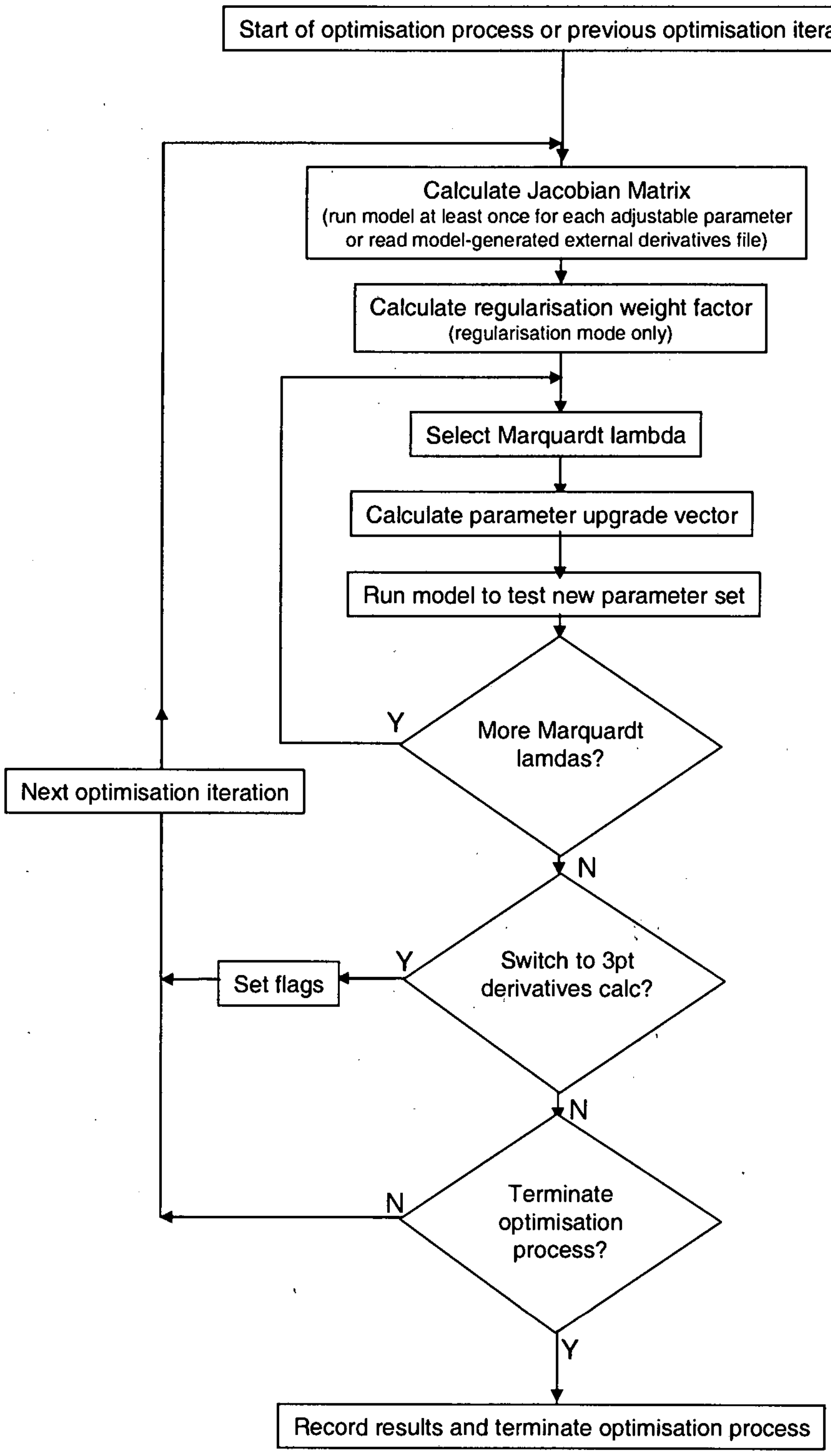
Start of optimisation process or previous optimisation iteration
Calculate Jacobian Matrix
(run model at least once for each adjustable parameter or read model-generated external derivatives file)
Calculate regularisation weight factor
(regularisation mode only)
Select Marquardt lambda
Calculate parameter upgrade vector
Run model to test new parameter set
More Marquardt lamdas?
Y
N
Next optimisation iteration
Switch to 3pt derivatives calc?
Y
Set flags
N
Terminate optimisation process?
N
Y
Record results and terminate optimisation process

**Figure 3. Structure of an individual optimization iteration.**

## 3.8 PEST-Model Interface

One of the cornerstones of PEST's model-independence will be its ability to communicate with a model through the model's own input and output files. Thus whenever it runs the model (whether it be for the calculation of derivatives with respect to a particular parameter, to calculate derivatives with respect to all parameters, or to test the effects of an upgraded parameter set on model outputs), PEST will first write a set of model input files containing the parameter values that it wishes the model to use on that particular run. Once the model has finished running, PEST will read those numbers from the model's output files for which there are corresponding field or laboratory measurements.

The design of the PEST model-interface is now discussed. Note that the discussion in this section is restricted to communication that takes place between PEST and a model through the model's own input and output files. More sophisticated (and optional) means of communication involving the use of a PEST-to-model message file and/or an external derivatives file are discussed in Section 3.3 of this document.

### 3.8.1 Model Input Files

PEST will write model input files on the basis of "templates" of these files prepared by the user. Each such template file will be prepared by modification of a corresponding model input file. In each of these template files, the user will identify the locations on the corresponding model input file at which numbers representing adjustable parameters reside. In construction of the template file, the user will replace these numbers by character strings that will convey to PEST both the name of the adjustable parameter, and the space to which the value of that parameter must be written when PEST prepares the model input file on the basis of the template file.

To illustrate the manner in which a template file is prepared from a model input file, Figure 4 shows a model input file while Figure 5 shows a template file prepared from the model input file.

```
MODEL INPUT FILE
3, 19                           no. of layers, no. of spacings
1.0, 1.0                              resistivity, thickness: layer 1
40.0, 20.0                      resistivity, thickness: layer 2
5.0                             resistivity: layer 3
1.0                             electrode spacings
1.47
2.15
3.16
4.64
6.81
10.0
14.9
21.5
31.6
46.4
68.1
100
149
215
316
464
681
1000
```

**Figure 4. A model input file.**

```
ptf #
MODEL INPUT FILE
3, 19                                    no. of layers, no. of spacings
#res1    #,#t1      #                    resistivity, thickness: layer 1
#res2    #,#t2      #                    resistivity, thickness: layer 2
#res3    #                               resistivity: layer 3
1.0                                      electrode spacings
1.47
2.15
3.16
4.64
6.81
10.0
14.9
21.5
31.6
46.4
68.1
100
149
215
316
464
681
1000
```

**Figure 5. Template file corresponding to model input file of Figure 4.**

The following design characteristics of a template file are apparent from a comparison of Figures 4 and 5.

- The template file will begin with the characters "ptf" (for "PEST template file"), followed by a space, followed by a single character which will be referred to as the "parameter space delimiter". This character will be used in the remainder of the file to indicate to PEST the locations to which current parameter values must be written when it writes the model input file. This character will be user-selectable; however the user should select a character that is used nowhere else on the model input file (a character such as "~", "#", "^", "@", etc. will be suitable on most occasions).

- The locations on the template file where the current values of the adjustable parameters are to be written when creating the model input file are referred to as "parameter spaces". A parameter space must begin and end with the parameter space delimiter. Between these delimiters is a string (of up to 12 characters in length) denoting the name of the parameter.

Prior to running the model, PEST will replace each parameter space with a number representing the current value of the identified parameter. It will be the user's responsibility when preparing a template file from a model input file to ensure that the model's input formatting requirements are respected when PEST replaces the parameter space with the pertinent parameter value. PEST, for its part, will take the following steps to ensure that numbers are written to model input files in a way that best serves the purposes of the optimization process:-

- Depending on the setting of the PEST input variable DPOINT, PEST will omit the decimal point from its representation of current parameter values if this can be done without changing the value of these parameters. This will allow an extra significant figure to be used in the representation of the number if the formatting requirements of the model input file are such that the space to which the number can be written is limited. In certain circumstances, the extra significant figure may allow a particular parameter value to be numerically distinguishable from its value when incremented for the purposes of derivatives calculation.

- If the PEST input variable PRECIS is set in an appropriate manner, PEST will represent parameter values using double precision protocol (if the width of parameter spaces allows it).

- Before PEST writes a parameter value to a model input file, it will first write it to an internal character variable using exactly the same ASCII representation of this number as that which it will use when writing that number to the model input file. It will then re-read that number from the character variable in order to ensure that the internal and external representations of that number are identical. This will increase accuracy in derivatives calculation when undertaken on the basis of finite parameter differences.

There will be no internal limit on the number of model input files that PEST will be capable of writing on the basis of corresponding template files prior to running the model.

### 3.8.2 Model Output Files

Unfortunately, it will not be possible to employ the template concept in the reading of model output files, for these can change from model run to model run. Hence PEST will read model output files by implementing a set of instructions supplied by the user. For a given model output file, the instruction set will direct PEST to those numbers for which there are complementary field or laboratory measurements.

Instructions will need to be prepared by the user prior to a PEST run. An "instruction file" will be required for each model output file from which PEST is required to read one or more numbers. An example of a model output file and a corresponding instruction file are provided in Figures 6 and 7.

```
SCHLUMBERGER ELECTRIC SOUNDING

Apparent resistivities calculated using the linear filter method

electrode spacing  apparent resistivity
    1.00             1.21072
    1.47             1.51313
    2.15             2.07536
    3.16             2.95097
    4.64             4.19023
    6.81             5.87513
    10.0             8.08115
    14.7             10.8029
    21.5             13.8229
    31.6             16.5158
    46.4             17.7689
    68.1             16.4943
    100.             12.8532
    147.             8.79979
    215.             6.30746
    316.             5.40524
    464.             5.15234
    681.             5.06595
   1000.             5.02980
```

**Figure 6. A model output file.**

```
pif @
@electrode@
l1 [ar1]21:27
l1 [ar2]21:27
l1 [ar3]21:27
l1 [ar4]21:27
l1 [ar5]21:27
l1 [ar6]21:27
l1 [ar7]21:27
l1 [ar8]21:27
l1 [ar9]21:27
l1 [ar10]21:27
l1 [ar11]21:27
l1 [ar12]21:27
l1 [ar13]21:27
l1 [ar14]21:27
l1 [ar15]21:27
l1 [ar16]21:27
l1 [ar17]21:27
l1 [ar18]21:27
l1 [ar19]21:27
```

**Figure 7. An instruction file built to read apparent resistivity values from the model output file depicted in Figure 6.**

The PEST instruction set will include the following functionality:-

- It will allow PEST to peruse a model output file, line by line, looking for a key character string. The location of this string will then serve as a reference point for further perusal of the file.
- It will allow PEST to advance forward in a file by a user-specified number of lines.
- It will allow key character strings to be used for navigation purposes within an individual line of a model output file.
- It will allow whitespace and arbitrary numbers generated by the model to serve as in-line navigation points.

A PEST instruction file will always begin with the letters "pif" followed by a user-supplied character, the "marker delimiter"; the marker delimiter will be used to delimit character strings in instructions that search for such strings. These strings will be referred to as "markers" because of their role in providing a reference point for further reading of a model output file. The "electrode" string depicted in Figure 7 is an example of such a character string; according to the "@electrode@" instruction, PEST will read the pertinent model output file until it finds a line containing the string "electrode". It will then implement the next instruction. The marker delimiter character can be any character that is not found in any of the character strings for which a search must be made.

Once a number within a model output file for which there exists a corresponding field or laboratory measurement has been located by the "navigational component" of the instruction set, the number will be read in one of three ways using one of three different types of instruction. In each case the number will be named as it is read so that it can be linked to a measurement value recorded in the PEST control file (see later). The three ways in which a number will be read from the model output file will be as follows:-

- through demarcation of the character positions within the model output file between which the number lies;
- through demarcation of the approximate character positions between which the number lies (PEST will then define the exact extent of the number itself);
- through definition of the number's location as the next non-whitespace string following a particular navigational item.

Table 3 provides the list of instructions which will comprise the PEST instruction set, together with a brief description of the role of each.

| Instruction | Example | Operation |
|---|---|---|
| Line advance | L3 | Advance through the model output file by the supplied number of lines. |
| Primary marker | $FLOW$ | Advance through the model output file line-by-line until the supplied string is found. |
| Secondary marker | $FLOW$ | Advance along a particular line of the model output file until the supplied string is found. |
| w | whitespace | Advance along a line until a tab or space is found. |
| !dum! | dummy observation | Advance to the end of the next number occurring on a line of the model output file. |
| tab | t32 | Advance to the indicated character position on a line of the model output file. |
| fixed observation | [obs1]13:30 | Read the named model output from between the designated character positions on the current line of the model output file. |
| semi-fixed observation | (obs1)20:30 | Read the named model output from between the designated character positions; however the width of the character interval can be expanded if necessary. |
| non-fixed observation | !obs1! | Read the named model output as the next number on the current line of the model output file. |

**Table 3. List of instructions to be used by PEST.**

Two types of marker are referenced in Table 3. Where any line of an instruction file begins with a marker, then PEST will read the model output file until it finds a line containing the marker string; this type of marker is referred to as a "primary marker". However where a marker follows other instructions within a single instruction line, then that marker is used for navigation within that line only; PEST will not read further lines of the model output file to find the marker. This type of marker is referred to as a "secondary marker".

There will be no limit to the number of model output files that PEST will be capable of reading on the basis of corresponding instruction files after it has run the model.

## 3.9 Restarting a PEST Run

### 3.9.1 Data Storage

At the end of every optimization iteration PEST will record the Jacobian matrix on a binary file (the "Jacobian matrix file"). It will also store, in another binary file (the "restart file"), the current values of all variables involved in the optimization process.

### 3.9.2 Restarting at the Beginning of the Current Optimization Iteration

If PEST execution is interrupted, PEST will be capable of being re-started using a special command-line switch (i.e. "/r") that will instruct it to re-commence execution at the beginning of the optimization iteration at which its execution was interrupted. When re-started using this switch, PEST will read some of its data through its normal input files, but will then read the binary "restart file" for the remainder of its data; this data will include its current optimization iteration number, current parameter values, and much of the history of the optimization process up until the time of interruption of execution. PEST will then re-commence execution at the beginning of the current optimization iteration as read from the restart file. Note, however, that when re-started in this manner it will not read the Jacobian matrix file recorded on the previous PEST run. Hence it will not be able to recommence execution at any place other than at the beginning of the latest optimization iteration.

### 3.9.3 Restarting at the Location of Latest Jacobian Calculation

If restarted with another command-line switch (i.e. the "/j" switch), PEST will re-commence execution of a previous run at the point at which it had last finished calculation of the Jacobian matrix. If its execution was previously interrupted while testing the effects of different Marquardt lambdas on the objective function, it will thus re-commence execution in the same optimization iteration as that in which its previous execution was interrupted. However if, when it was previously halted, it was undertaking model runs for the purpose of filling the Jacobian matrix, then re-commencement of execution at that point at which it had most recently filled the Jacobian matrix will take it back to the previous optimization iteration.

When re-started using the "/j" switch, PEST will read the binary Jacobian matrix file, as well as the restart file recorded during the previous run. If PEST ascertains that it must re-visit the previous optimization iteration, it will then read the restart file pertaining to the previous optimization iteration. (On commencement of each new optimization iteration, PEST will copy its existing restart file to a "previous iteration restart file" before overwriting it with the latest restart file, so that, if/when execution is re-commenced, restart files from both the current and previous optimization iterations will be available for the use of the re-started PEST.)

Note that a user will be able to obtain a copy of the Jacobian matrix in ASCII format for his/her own inspection by running the PEST JACWRIT utility; see section 3.18.7 for further details.

## 3.10 Parameter Change Limits

PEST will calculate the parameter upgrade vector using either of equations (1) or (2), the appropriate equation depending on its mode of operation. However a user-supplied limit will be imposed on the amount by which any parameter will be allowed to change during any one optimization iteration. Where this limit must be enforced by PEST because the change to at least one parameter, as recorded in the parameter upgrade vector, exceeds the user-imposed limit, the magnitude of the parameter upgrade vector will be reduced, but its direction will be maintained.

Parameter change limits will be of two types - relative and factor; the maximum relative parameter change will be supplied through the input variable RELPARMAX while the maximum factor change will be supplied through the input variable FACPARMAX. Each parameter will then be designated as either "relative-limited" or "factor-limited" (through the PEST input variable PARCHGLIM). The user-supplied relative limit (i.e. RELPARPMAX) will apply to all relative-limited parameters. The user-supplied factor limit (i.e. FACPARMAX) will apply to all factor-limited parameters. If a parameter is designated as relative-limited, then the magnitude of its change relative to its current value will be limited. If it is designated as factor-limited, then the ratio of its new value to its old value (or the reciprocal of this) will be subject to the user-supplied factor limit.

## 3.11 User-Intervention Functionality

### 3.11.1 Design Considerations

The most time-consuming part of the optimization process undertaken by PEST will be computation of the Jacobian matrix, for PEST will need to run the model at least as many times as there are parameters to be estimated. Where model run-times are large, this may require a considerable amount of CPU time. Even when employing PEST's external derivatives functionality whereby the model calculates derivatives itself, the calculation of these derivatives is still likely to be a CPU-intensive procedure.

Once the Jacobian matrix has been filled, PEST will calculate different parameter upgrade vectors using equation (1) or (2) based on different values of the Marquardt lambda. In each of equations (1) and (2), the matrix that PEST must invert to calculate the parameter upgrade vector is referred to as the "normal matrix". If PEST encounters any difficulties in inverting this matrix, it will automatically raise the Marquardt lambda (thus ensuring a degree of diagonal dominance), and attempt the upgrade calculation again.

If one or a number of particular parameters are more insensitive than the rest, or if two or more parameters are very highly correlated with each other (meaning that they can be varied together in a certain relationship with almost no effect on the objective function), then it is possible that the normal matrix will be nearly singular. This condition will lead to gross amplification of numerical errors, and hence to computation of a parameter upgrade vector which may be far from optimal. Parameter insensitivity can also lead to a situation where one of two relatively insensitive parameters need to change by a large amount to incur any effect on the objective function. PEST may thus calculate large changes for these parameters in the parameter upgrade vector. However by limiting changes to these parameters to those enforced by the maximum factor and relative change limits, while at the same time maintaining the direction of the parameter upgrade vector, sensitive parameters (the parameters whose effects on the objective function are greatest), may hardly change at all. Hence, during any one optimization iteration, the objective function may improve very little. PEST may then "trip" into three-point derivatives calculation earlier than it should (if derivatives are calculated using finite parameter differences), further lengthening the overall optimization process.

PEST will write to the screen, to its parameter sensitivity file (see below), to its matrix file (see below), and to its run record file (see below), enough information for the user to be able to judge whether calculation of the parameter upgrade vector could have been improved if one or a number of insensitive or highly correlated parameters were temporarily held fixed. Such information will include:-

- Marquardt lambda values used in the calculation of parameter upgrade vectors (if PEST raises the Marquardt lambda rather than lowers it, the user can infer that PEST is having difficulties in inverting the normal matrix),
- objective function values resulting from the use of different trial Marquardt lambdas,
- composite sensitivities of all adjustable parameters (see below),
- the parameter covariance and correlation coefficient matrices, as well as the eigenvalues/eigenvectors of the covariance matrix, all based on current parameter values (see below),
- the name of the parameter that underwent the largest relative or factor change.

If, through a perusal of this information, the user suspects that computation of the parameter upgrade vector was hampered by the presence of one or more insensitive or correlated parameters, he/she will have the option of halting PEST execution, holding the offending parameter(s) fixed at their current value(s), and re-commencing the parameter estimation process at that point at which the Jacobian matrix was last filled. Thus the most time-consuming component of each optimization iteration (viz. calculation of the Jacobian matrix) will not need to be undertaken again. If it is then found that other parameters need to be held, the stopping and restarting process can be repeated, with the extra parameters held at their current values.

### 3.11.2 The Parameter Hold File

At the commencement of each optimization iteration PEST will look in its current working directory for a "parameter hold file". This will be an ASCII file, easily written by the user using a text editor. Lines within this file will be able to command PEST to do any of the following:-

- hold an individual parameter (identified by its name) at its current value,
- hold parameters whose composite sensitivities fall below a user-supplied threshold at their current values,
- hold the *n* parameters of least sensitivity within a particular parameter group at their current values, where *n* is supplied by the user,
- hold the parameters responsible for the *n* largest components of the eigenvector corresponding to the *m*'th largest eigenvalue at their current values, where *n* and *m* are supplied by the user, and
- alter the values of a number of PEST control variables (viz. RELPARMAX, FACPARMAX and LAMBDA, these being the relative and factor change limits and the current value of the Marquardt lambda respectively).

Figure 8 shows an example of a parameter hold file while Figure 9 shows a flow chart of the PEST user intervention process.

```
relparmax 10.0
facparmax 10.0
lambda 200.0
hold parameter thick1
# hold parameter thick2
hold group conduct < 15.0
hold group thicknss lowest 3
hold eigenvector 1 highest 2
```

**Figure 8. Part of a parameter hold file.**

Full details of the syntax required for commands issued through the parameter hold file will be provided in the User's Manual.

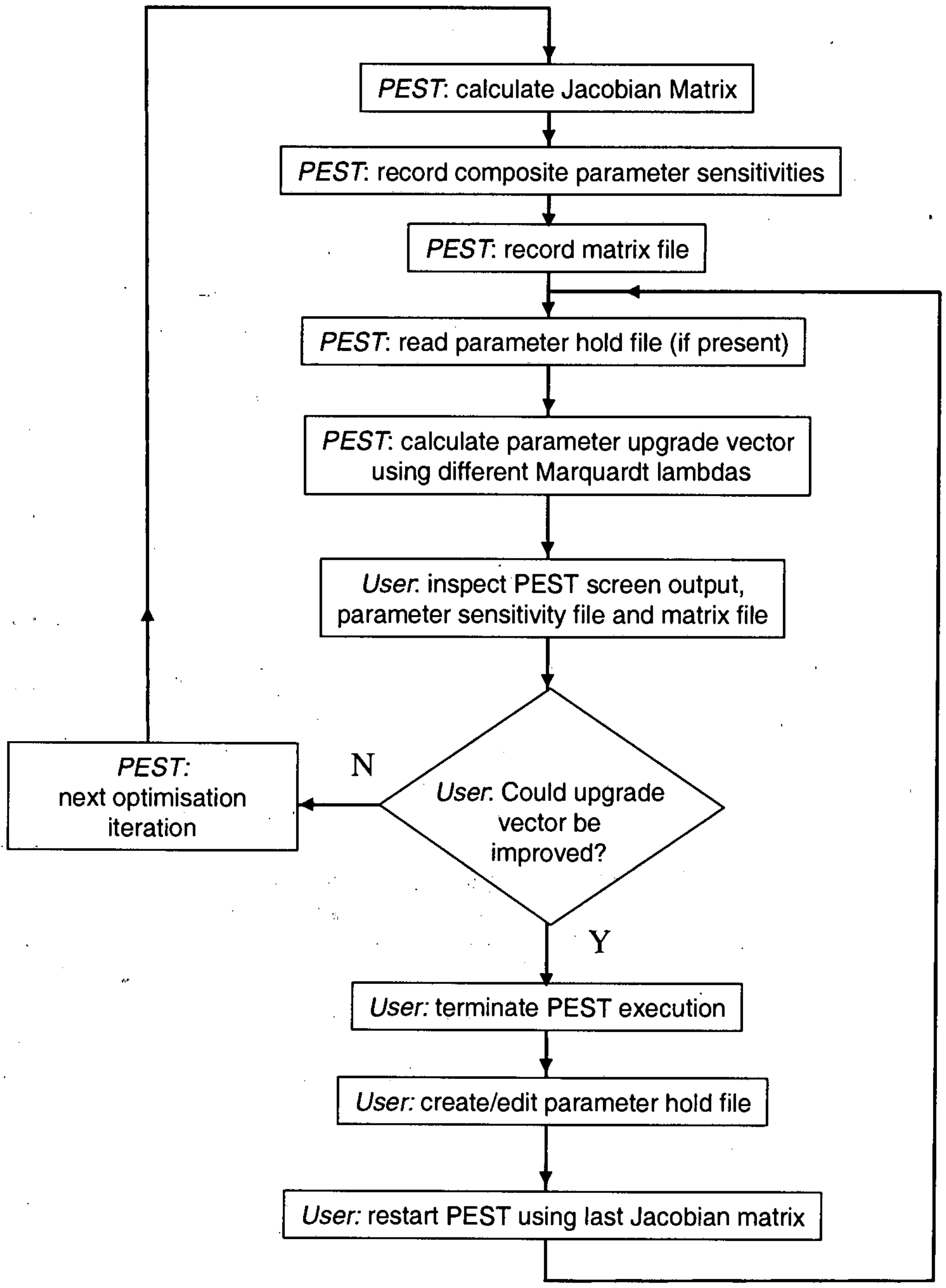
PEST: calculate Jacobian Matrix
PEST: record composite parameter sensitivities
PEST: record matrix file
PEST: read parameter hold file (if present)
PEST: calculate parameter upgrade vector using different Marquardt lambdas
User: inspect PEST screen output, parameter sensitivity file and matrix file
User: Could upgrade vector be improved?
N
PEST: next optimisation iteration
Y
User: terminate PEST execution
User: create/edit parameter hold file
User: restart PEST using last Jacobian matrix

**Figure 9. Flow chart of the PEST user-intervention process.**

## 3.12 Parameter Transformations and Linkages

### 3.12.1 General

Parameters defined in template files will be designated as either adjustable, fixed or tied in the PEST control file using the PEST input variable PARTRANS. Adjustable parameters will be further designated as undergoing log transformation through the parameter estimation process, or as undergoing no transformation.

### 3.12.2 Log Transformation

If a parameter is log-transformed PEST will estimate the log (to base 10) of that parameter rather than the parameter itself. Thus the Jacobian matrix will contain derivatives with respect to the log of that parameter, and the element of the parameter upgrade vector calculated by PEST pertaining to that parameter will actually contain the alteration to the log of the parameter value.

### 3.12.3 Fixed Parameters

If a parameter is designated as "fixed", its value will not be altered throughout the parameter estimation process.

### 3.12.4 Tied Parameters

If a parameter is designated as "tied", then it will not be estimated. Rather it will be varied throughout the parameter estimation process in such a way that the ratio of its current value to that of its "parent parameter" (i.e. the parameter to which it is tied) is maintained. The parent parameter must be an adjustable parameter. Calculation of derivatives with respect to the parent parameter will take into account the fact that at least one other parameter is tied to it. Hence when it is incremented for the purposes of derivatives calculation, so too will be its "child parameter(s)".

## 3.13 Parameter Bounds

PEST will require that the user supply an upper and lower bound for each adjustable parameter. PEST will ensure that no parameter exceeds its upper bound or becomes less than its lower bound at any stage of the optimization process. Bounds will be supplied through the PEST input variables PARLBND and PARUBND.

If a parameter upgrade vector **u** calculated through equation (1) or (2) is such as to cause one or more parameters to move beyond their limits, PEST will adjust **u** such that this does not occur, placing such parameters at their upper or lower bounds. During the next optimization iteration, special treatment will be afforded to these parameters. If the component of both the parameter upgrade vector and the negative of the objective function gradient vector pertaining to a parameter at its upper or lower limit are such as to take the parameter out of bounds, then the parameter will be temporarily frozen, and the optimization problem reformulated with that parameter fixed at its limit; hence the new upgrade vector will not result in any adjustment to that parameter. If, after reformulation of the problem in this manner, with all such parameters temporarily held fixed, there are parameters at their limits for which the parameter upgrade vector still points outward from the allowed parameter domain, while the negative of the gradient vector points inward, then these parameters, too, will be temporarily frozen. This process continues until a parameter upgrade vector is calculated which either moves parameters from their bounds back into the allowed parameter domain, or leaves them fixed.

The strength of this strategy is that it will allow PEST to search along the boundaries of the allowed parameter domain for the smallest objective function to which it has access when the global minimum of the objective function actually lies outside of the allowed parameter domain, beyond PEST's reach.

At the beginning of each new optimization iteration all temporarily-frozen parameters will be freed to allow them to move back inside the allowed parameter domain if solution of equation (1) or (2) deems this necessary. If not, the stepwise, temporary freezing of parameters will then be repeated as described above.

## 3.14 Prior Information

PEST will allow the user to include "prior information" in the optimization process in the form of preferred values for one or a number of adjustable parameters, or of linear relationships between these parameters. If a parameter is log-transformed, any prior information supplied for that parameter must pertain to the log of that parameter.

Mathematically, prior information will be incorporated into the optimization process as a set of "observations". Differences between preferred values of the prior information equations and those calculated on the basis of current parameter values will thus constitute an extra set of residuals which will then be combined with the model-to-measurement residuals to compute the overall objective function whose task it will be for PEST to minimize. Derivatives of the prior information equations with respect to adjustable parameters will be included in the Jacobian matrix for computation of the parameter upgrade vector using equation (1) or (2). However, due to the linear nature of each prior information equation, pertinent elements of the Jacobian matrix will not require calculation through finite differences or need to be supplied by the model. Rather they will be equal to the parameter factors (PEST variable PIFAC) cited in the prior information equations.

The user will be required to assign a weight (see below) to each prior information equation, this governing its contribution to the objective function, and hence the influence which it has on the optimization process in comparison with other prior information equations and with members of the field or laboratory data set. Each article of prior information must be assigned to an observation group (see below). This will allow the contribution to the objective function by different prior information equations (or groups of prior information equations) to be monitored during the optimization process. Through assigning prior information equations to the special observation group "regul", prior information can contribute to the regularization process.

Each prior information equation will need to be assigned a name by the user (PEST variable PILBL) of 12 characters or less in length.

## 3.15 Observations

### 3.15.1 General

As has already been discussed, PEST will calculate an "objective function", formulated as the weighted sum of squared differences between model outputs and corresponding field or laboratory measurements; each such difference is referred to as a "residual". When working in parameter estimation mode or regularization mode, PEST's task will be to reduce the objective function as far as possible while maintaining all parameters within their bounds. When working in predictive analysis mode, PEST will be asked to maximize or minimize a key model prediction, while maintaining the objective function at or below a user-specified threshold (PEST input variable PD1).

### 3.15.2 Observation Names

Each observation must be assigned a name by the user (PEST variable OBSNME) of 12 characters or less in length.

### 3.15.3 Observation Groups

Each observation must be assigned to an observation group (PEST input variable OBGNME). Use of these groups will allow PEST to report to its output file the collective contribution made to the objective function by all observations and/or prior information items belonging to the group.

### 3.15.4 Observation Weights

The objective function Φ will be calculated using the equation:-

$$\Phi = \sum w_i r_i$$

where:-

| | |
|---|---|
| $r_i$ | is the $i^{th}$ residual, and |
| $w_i$ | is the weight assigned to the $i^{th}$ observation. |

Summation will take place over all observations and over all elements of prior information. Where PEST evaluates the contribution made to the objective function from different groups, summation will also take place over each group individually; PEST will then report each group sub-total to its run record file.

### 3.15.5 Observation Covariance Matrix

The use of observation weights in calculating the objective function is based on the premise that observations are independent, i.e. that the "uncertainty" pertaining to any one observation bears no relationship to the "uncertainty" pertaining to any other observation. In practice, observation "uncertainty" in a calibration context is determined by the level of misfit between these observations and corresponding model outputs, i.e. by the model-to-measurement residuals calculated at the end of the inversion process. If these residuals are expected to be uncorrelated (as will occur in the majority of parameter estimation contexts), then observation uncertainties can indeed be expressed in terms of individual observation weights. However if residuals are likely to show consistency over space and/or time for certain observation types, then it may not be appropriate to assume statistical independence for these observation types. In such cases it may be preferable to describe the uncertainties associated with these observations using an observation covariance matrix (or a matrix that is proportional to this matrix), rather than using a set of individual observation weights.

PEST will allow the user the option of supplying an observation covariance matrix for one or more user-defined observation groups. When using such a matrix the theory upon which PEST's operations are based (as expressed by equations 1 and 2) remains unchanged if the inverse of the observation covariance matrix is substituted for **Q** (or for the subset of **Q** represented by the pertinent observation group). The observation covariance matrix pertaining to each observation group must be supplied to PEST in an ASCII file, the name of this file being recorded in the "observation groups" section of the PEST control file (PEST variable COVFILE) adjacent to the observation group name to which it corresponds. Figure 10 shows an example of a covariance matrix file.

```
1.0  0.1  0.0  0.0
0.1  1.0  0.1  0.0
0.0  0.1  1.0  0.1
0.0  0.0  0.1  1.0
```

**Figure 10. Example of an observation covariance matrix file.**

Theoretically, the observation covariance matrix file must be positive definite, and hence symmetric. However PEST will be capable of accepting such a file as long as it is symmetric.

### 3.15.6 Predictive Analysis

As explained in Section 3.1.1, when used in predictive analysis mode PEST's task will be to maximize or minimize one particular model outcome while maintaining the objective function (defined on the basis of all other model outcomes) below a user-defined threshold; this particular model outcome will be referred to as the "prediction". Like other model outcomes of interest, it will be read (using the pertinent instruction) from a model output file; and like other model outcomes of interest it will have a name of twelve characters or less in length. However it will be distinguished from other observations by being assigned to the observation group "predict" as the sole member of this group.

### 3.15.7 Regularization

When PEST operates in regularization mode, at least one observation must be assigned to the observation group "regul". Weights (or a covariance matrix) must be assigned to observations comprising the members of this group in the same way that they must be assigned to members of other observation groups. In many cases of practical interest the members of this group will not correspond to actual field observations; instead they will express preferred values for relationships between parameters based on smoothness or geostatistical assumptions. Such relationships (if they are linear) can also be provided as prior information equations, distinguished from other elements of prior information through being assigned to the observation group "regul".

During each optimization iteration, PEST will compute a "regularization weight factor". The weights (or the inverse of the covariance matrix) assigned to all members of the observation group "regul" will be multiplied by this factor in computation of the overall objective function. The value of this factor will be such that, under the linearity assumption employed during each optimization iteration, the "measurement objective function" (i.e. the component of the objective function computed using all observations other than those assigned to the observation group "regul") will be equal to the user-assigned upper limit for this component of the objective function (i.e. PEST input variable PHIMLIM). As the "measurement objective function" will normally be calculated on the basis of the discrepancies between field observations and their model-generated counterparts, calculation of the weight factor in this manner will ensure that an upper limit is placed on the allowed misfit between model outputs and field data used in the calibration process when attempts are made to simultaneously satisfy smoothness or geostatistical constraints on spatial parameter distributions.

As it is not possible to derive an analytical expression for the weight factor, computation of the weight factor will take place through an iterative procedure that will be deemed to reach completion upon satisfaction of a user-supplied convergence criterion (PEST input variable WFTOL).

If the user is unable to assign an appropriate value to the PEST input variable PHIMLIM when preparing for a PEST run, he/she will be able to request that PEST calculate a suitable value for this variable itself from the current value of the objective function, and adjust this value as the objective function falls. At any stage of the regularized parameter estimation process, PEST will calculate this temporary value of PHIMLIM by multiplying the current value of the measurement objective function by the user-supplied value for FRACPHIM. Ideally, FRACPHIM should be between 0.2 and 0.3. In this way, the numerical benefits of regularization will be retained, even where determination of a suitable value for PHIMLIM in advance of the parameter estimation process is problematical.

## 3.16 Precision

All calculations performed by PEST will take place in double precision.

## 3.17 Parallel Processing

### 3.17.1 Design Considerations

The most time-consuming part of the parameter estimation process will be the filling of the Jacobian matrix. As is explained above, when model derivatives are calculated by finite parameter differences, PEST will need to run the model at least as many times as there are adjustable parameters. During each such model run, one parameter will be incrementally varied from its current value. Derivatives with respect to that parameter will be calculated using a two-point or three-point finite difference scheme.

The calculation of derivatives will be an ideal candidate for parallelization due to the fact that each model run required to fill a column of the Jacobian matrix will be independent of model runs required to fill other columns of the Jacobian matrix. That is, the parameters supplied to the model for one of these runs do not depend on the outcomes of any of the other model runs.

There will also be potential for gains in efficiency to be made if the Marquardt lambda testing process is "partially parallelized". Although the choice of what lambda to use on a particular model run will depend on the results of previous model runs undertaken on the basis of other lambda values, any lambda value used by the model will be displaced from the original lambda value used in this process by one or a number of multiples of the PEST input control variable LAMFAC. Thus if networked computers and/or alternative processors are standing idle, nothing will be lost, and potentially much will be gained, if these machines and/or processors are engaged in undertaking model runs on the basis of lambda values which are related to the current lambda value by factors of LAMFAC. Implementation of such a "partial parallelization" process can result in significant gains in efficiency, though the gains will not be quite as spectacular as those to be had through parallelization of the finite-difference derivatives calculation process as the results of some of these lambda-testing runs will never be used by PEST.

### 3.17.2 Slave Programs

PEST will allow parallelization of model runs through the agency of a number of "slave programs" running either on the same machine as that on which PEST is running, or on different machines to which PEST has access through a local area network. Each of these slave programs (named PSLAVE) will be started up (by the user) in a different working directory, the name of this directory being provided to PEST in its "run management file" (see below). On reception of an appropriate command from PEST, each slave program will run the model. However before it issues the command to a particular slave to run the model, PEST will write model input files containing pertinent parameter values to the slave's working directory, using template files residing in its own working directory. At the completion of each model run, PEST will read pertinent outcomes from model output files residing in the pertinent slave's working directory using instruction files residing in its own working directory.

For this parallelization mechanism to work correctly, the following conditions must be fulfilled:-

- Each slave must be capable of running the model from the machine on which it resides; in many cases this will most easily be achieved by placing a copy of the model executable on each slave's machine.
- Each slave must use a different working directory so that model input and output files produced by the model when run under the supervision of one particular slave are not overwritten by the model when run under the supervision of another slave.
- The working directory of each slave must be "visible" to PEST from the machine and directory on which it is running.
- All model input files which do not contain adjustable parameters (and hence which are not re-written by PEST prior to each model run) must be accessible by the model when run by the slave. (In most cases this condition will be satisfied if copies of all such files are placed within the working directories of the various slaves.)

Figure 11 schematizes the relationship that will exist between PEST and its slaves.

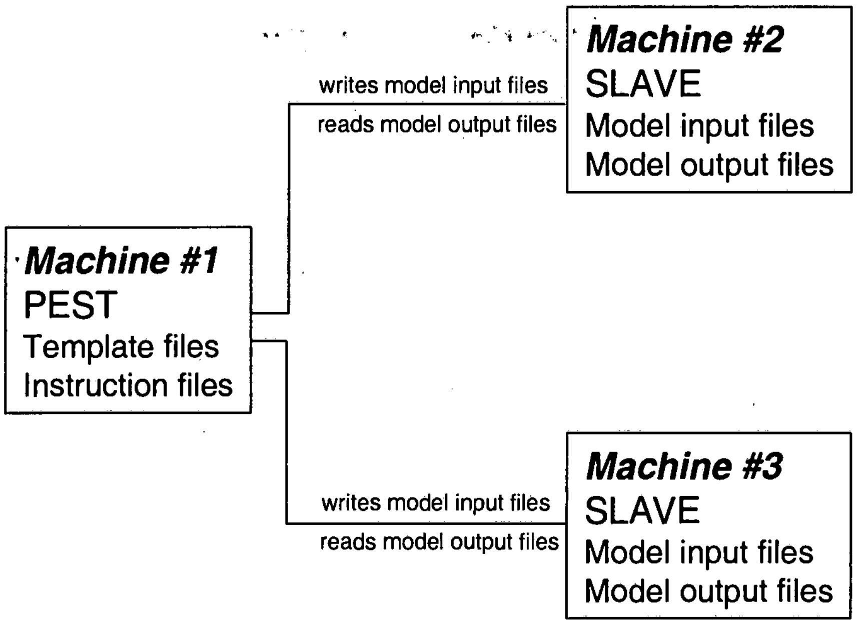

**Figure 11. Schematic of the relationship between PEST and its slaves.**

Note that Figure 11 illustrates a case where three machines are involved in the parallelization process and the slaves are run on different machines from that on which PEST is run. In many cases at least one slave will reside on the same machine as PEST. (PEST's computational burden will usually be light compared with that of the model as run by the slave.) Furthermore, where a machine has multiple CPU's, it may be efficient to operate as many slaves on that machine as there are CPU's in order to take maximum advantage of the processing capabilities offered by that machine.

### 3.17.3 Communication between PEST and its Slaves

To maintain generality across all computing platforms, and in order to avoid the use of third party software, PEST will communicate with its slaves using simple "semaphore files". The names and roles of each of these files will be as illustrated in Table 4.

41

| File | Written By | Function |
|---|---|---|
| *pslave.rdy* | PSLAVE | Informs PEST that it has begun execution; also informs it of the command which it will use to run the model. |
| *pest.rdy* | PEST | Informs PSLAVE that it has begun execution. |
| *param.rdy* | PEST | Informs PSLAVE that it has just generated model input files(s) on the basis of a certain parameter set and that it must now run the model. |
| *observ.rdy* | PSLAVE | Informs PEST that the model has finished execution and that it must now read the model output file(s). |
| *pslave.fin* | PEST | Informs PSLAVE that it must now cease execution. |
| *p###.##* | PEST | Used to test whether PEST has access to all PSLAVE working directories. |

**Table 4. Files used by PEST and PSLAVE to communicate with each other.**

### 3.17.4 The Slave Program

Like PEST, the slave program PSLAVE will be written in FORTRAN, maintaining the same programming standards as that outlined in Section 2.1 for PEST. Hence it will be possible to re-compile this program for any platform using whatever FORTRAN compiler is available for that platform. The only non-standard item of functionality involved in the programming of PSLAVE will be the system call which PSLAVE will use to run the model. Fortunately, though not a FORTRAN standard, the protocol for a system call is similar across most commonly-used compilers. Where there are differences, compiler directives can be used to select the appropriate code segment.

PSLAVE source code and makefile (optimized for the UNIX environment) will be provided with PEST. An executable version of PSLAVE, compiled for use on a PC running WINDOWS-2000, will also be provided.

## 3.18 Utility and Checking Programs

### 3.18.1 Design Considerations

A number of utility programs will be provided with PEST in order to facilitate the construction of some of its input files, to check that the entire input data set supplied to PEST for any optimization run is consistent and correct, and to undertake various pre- and post-processing tasks. Like PEST, all of these programs will be written in near-standard FORTRAN and will thus be easily compiled for use on any system for which a FORTRAN compiler is available. Executable versions of these utilities will be supplied for PCs running the WINDOWS 2000 operating system. A description of the role of each utility is now provided.

### 3.18.2 TEMPCHEK

TEMPCHEK will check PEST template files. Two levels of checking will be provided.

At its lowest level TEMPCHEK will check that a particular template file obeys the correct template syntax as set out in Section 3.8.1. The line number and description of any errors found in the template file will be written to the screen.

If supplied with the values of all parameters cited in a template file (through a "parameter value file" - see below), TEMPCHEK will write a model input file on the basis of the template file, thus emulating what PEST does prior to a model run. The user can thus check that the model input file has been written correctly. If desired, he/she can then run the model on the basis of the TEMPCHEK-generated model input file, the ultimate test of the veracity of a template file being that it can be used to generate a model input file that the model can read without error.

### 3.18.3 INSCHEK

INSCHEK will check PEST instruction files. Two levels of checking will be provided.

At its lowest level, INSCHEK will check that an instruction file is syntactically correct, reporting the locations and natures of any errors that it finds to the screen.

If provided with the name of a model output file, INSCHEK will read the model output file using the instruction set contained in the instruction file, recording all of the numbers which it reads from the model output file to an output file of its own (an "observation value file"). The user can thus verify that the instruction set contained within the instruction file is able to read the model output file in the manner for which it was designed.

### 3.18.4 PESTCHEK

PESTCHEK will read a PEST control file (see below) and all of the template and instruction files cited therein. It will check that the control file has the correct syntax and that the values for all PEST input variables provided in the control file are within the correct range and are consistent with each other. It will also check that all parameters cited in the control file are also cited in at least one of the template files listed in the PEST control file, and that all parameters cited in at least one of these template files are listed in the PEST control file. Similarly, it will ensure consistency of observation names between the PEST control file and all of the instruction files cited therein. It will report any errors or any inconsistencies that it encounters to the screen in a manner that will make rectification of the problem as simple as possible.

### 3.18.5 PESTGEN

PESTGEN will write a PEST control file based on a set of user-supplied parameter names and associated initial values contained within a parameter value file, together with the set of observation names and associated field or laboratory measurements contained within an observation value file. Default values (useable in most optimization contexts) will be supplied for all PEST control variables. In most instances a user will then be able to build his/her final PEST control file through limited alteration of the PESTGEN-generated PEST control file.

### 3.18.6 PARREP

PARREP will read a PEST control file and a parameter value file produced by PEST. It will generate a new PEST control file in which initial parameter values supplied in the original PEST control file are replaced by parameter values recorded in the parameter value file (the latter file normally containing optimized parameter values.). Thus it will be a trivial task to start a new PEST run using parameter values calculated by PEST during a previous run.

### 3.18.7 JACWRIT

JACWRIT will read a Jacobian matrix file written by PEST. This file is recorded by PEST in binary, rather than ASCII, format in order to save disk space in parameter estimation contexts in which the number of parameters and observations is large. JACWRIT will re-write this file in ASCII format for easy user inspection.

### 3.18.8 PAR2PAR

PAR2PAR will read an input file in which parameters are named and are assigned values using mathematical expressions of arbitrary complexity involving numbers and/or previously defined parameters. It will then record supplied and derived parameter values to one or a number of model input files. This will be achieved through the use of template files, for PAR2PAR's protocol for writing model input files based on supplied or derived parameter values will be identical to that of PEST.

Because of its ability to manipulate parameter values in accordance with user-defined expressions, PAR2PAR will find a useful role as a model pre-processor in a batch or script file run by PEST as a "composite model". A template can be constructed from a PAR2PAR input file. Prior to each composite model run, PEST will then provide PAR2PAR with a set of current parameter values. PAR2PAR will then use these parameter values to calculate the values pertaining to a "more advanced" parameter set for the use of the model. This will add considerable flexibility to the parameter estimation or predictive analysis process undertaken by PEST.

An example of a PAR2PAR input file is provided in Figure 12.

```
* parameter data
infilt1 = 0.3456
infiltrat2 = 1.0453
infiltrat3 = 1.5432
infilt2= infilt1 * infiltrat2
infilt3 = infilt2 * infiltrat3
* template and model input files
model1.tpl model1.in
model2.tpl model2.in
* control data
single point
```

**Figure 12. Example of a PAR2PAR input file.**

As the above example shows, parameters, and the relationships between parameters, will be defined in the "parameter data" section of a PAR2PAR input file. (In Figure 12 *infilt1*, *infiltrat2* and *infiltrat3* are the names of parameters.) The "template and model input files" section of the PAR2PAR input file will contain the names of template files, together with the names of the model input files to which they are linked. The "control data" section of the PAR2PAR input file will contain values for the variables PRECIS and DPOINT which will determine the protocol used for writing parameter values to model input files. These will have an identical role to their role in PEST (see Section 3.8.1).

Permissible operators in mathematical expressions to be used by PAR2PAR are listed in Table 5. Permissible functions are listed in Table 6.

| **Operator** | **Function of operator** |
|---|---|
| + | addition |
| - | subtraction |
| * | multiplication |
| / | division |
| ** | exponentiation |
| ^ | exponentiation |
| ( | right bracket |
| ) | left bracket |

**Table 5. Operators to be used in mathematical expressions provided in a PAR2PAR input file.**

| Function | Role of function |
|---|---|
| abs() | absolute value |
| acos() | inverse cosine |
| asin() | inverse sine |
| atan() | inverse tan |
| cos() | cosine |
| cosh() | hyperbolic cosine |
| exp() | exponential |
| log() | logarithm to base e |
| log10 | logarithm to base 10 |
| sin() | sine |
| sinh() | hyperbolic sine |
| sqrt() | square root |
| tan() | tan |
| tanh() | hyperbolic tan |

**Table 6. Mathematical functions to be supported by PAR2PAR.**

# 4. System Inputs and Outputs

## 4.1 PEST Input Files

PEST will require three types of input file, viz. template files, instruction files and a PEST control file. If one or more observation covariance matrices are supplied in lieu of some or all observation weights, then a set of files will also need to be supplied to PEST in each of which a covariance matrix will be housed. All of these files will be ASCII files which should be prepared prior to a PEST run by the user on the basis of detailed instructions to be found in the User's Manual (Watermark Numerical Computing, 2002). All of these files can be prepared using basic functionality encapsulated in a word processor or text editor. No inputs will be required from other existing systems, applications or instrumentation.

Template, instruction and observation covariance matrix files have already been discussed. The structure of the PEST control file is illustrated in Figure 13.

```
pcf
* control data
RSTFLE PESTMODE
NPAR NOBS NPARGP NPRIOR NOBSGP
NTPLFLE NINSFLE PRECIS DPOINT NUMCOM JACFILE MESSFILE
RLAMBDA1 RLAMFAC PHIRATSUF PHIREDLAM NUMLAM
RELPARMAX FACPARMAX FACORIG
PHIREDSWH
NOPTMAX PHIREDSTP NPHISTP NPHINORED RELPARSTP NRELPAR
ICOV ICOR IEIG
* parameter groups
PARGPNME INCTYP DERINC DERINCLB FORCEN DERINCMUL DERMTHD
(one such line for each of the NPARGP parameter groups)
* parameter data
PARNME PARTRANS PARCHGLIM PARVAL1 PARLBND PARUBND PARGP SCALE OFFSET DERCOM
(one such line for each of the NPAR parameters)
PARNME PARTIED
(one such line for each tied parameter; omit if no tied parameters)
* observation groups
OBGNME [COVFLE]
(one such line for each observation group)
* observation data
OBSNME OBSVAL WEIGHT OBGNME
(one such line for each of the NOBS observations)
* model command line
command to run model
(NUMCOM commands are required, listed one under the other)
* derivatives command line
command to run model in order for it to calculate parameter derivatives
EXTDERFLE
* model input/output
TEMPFLE INFLE
(one such line for each model input file containing parameters)
INSFLE OUTFLE
(one such line for each model output file containing observations)
* prior information
PILBL PIFAC * PARNME + PIFAC * log(PARNME) ... = PIVAL WEIGHT OBGNME
(one such line for each of the NPRIOR articles of prior information)
* predictive analysis
NPREDMAXMIN
PD0 PD1 PD2
ABSPREDLAM RELPREDLAM INITSCHFAC MULSCHFAC NSEARCH
ABSPREDSWH RELPREDSWH
NPREDNORED ABSPREDSTP RELPREDSTP NPREDSTP
* regularization
PHIMLIM  PHIMACCEPT FRACPHIM
WFINIT  WFMIN  WFMAX
WFFAC  WFTOL
```

**Figure 13. Construction details of the PEST control file.**

The PEST control file must begin with the character string “pcf”. It will then be divided into different sections, each section containing a series of variables that affect different aspects of PEST’s operations. Each section will begin with a header line which contains the name of the section preceded by the “*” character, exactly as shown in Figure 13. The “prior information”, “predictive analysis” and “regularization” sections can be omitted if they are not required. Note that a “predictive analysis” section and a “regularization” section should not reside in the same PEST control file, for PEST will not be able to run in both predictive analysis mode and regularization mode at the same time.

Within each section, the values of variables will be supplied in the order indicated in Figure 13; related variables are grouped together on the same line. Entries on each line should be space-delimited.

Details of the variables cited in each section of the PEST control file are provided in Tables 7 to 17. A more detailed description of the role of each variable, and of the range of reasonable values for each variable, will be provided in the PEST User's Manual (Watermark Numerical Computing, 2002).

| **Variable Name** | **Type** | **Values** | **Role of variable** |
|---|---|---|---|
| RSTFLE | character | "restart"<br>"norestart" | Determines whether PEST writes files containing restart information. |
| PESTMODE | character | "estimation"<br>"prediction"<br>"regularization" | PEST's mode of operation for current run. |
| NPAR | integer | > 0 | Number of parameters cited in "parameter data " section. |
| NOBS | integer | > 0 | Number of observation cited in "observation data" section. |
| NPARGP | integer | > 0 | Number of parameter groups cited in "parameter groups" section. |
| NPRIOR | integer | ≥ 0 | Number of articles of prior information cited in "prior information" section. |
| NOBSGP | integer | > 0 | Number of observation groups cited in "observation groups" section. |
| NTPFLE | integer | > 0 | Number of template files. |
| NINSFLE | integer | > 0 | Number of instruction files. |
| PRECIS | character | "single"<br>"double" | Governs numerical precision with which numbers are written to model input files. |
| DPOINT | character | "point"<br>"nopoint" | Determines whether the decimal point should be omitted from parameter values on model input files if possible. |
| NUMCOM | integer | > 1 | Number of commands used to run the model. |
| JACFILE | integer | 0 or 1 | 1 if model must supply external derivatives; 0 otherwise. |
| MESSFILE | integer | 0 or 1 | 1 if PEST must write a PEST-to-model message file; 0 otherwise. |
| RLAMBDA1 | real | > 0.0 | Initial value of Marquardt lambda |
| LAMFAC | real | > 1.0 | Marquardt lambda adjustment factor. |
| PHIRATSUF | real | > 0.0<br>< 1.0 | Sufficient ratio of new to old objective function to justify initiation of the next optimization iteration. |

| | | | |
|---|---|---|---|
| PHIREDLAM | real | > 0.0<br>< 0.5 | If relative objective function improvement between subsequent Marquardt lambdas is less than this, the next optimization iteration is initiated. |
| NUMLAM | integer | > 0 | Maximum number of Marquardt lambdas to test on any one optimization iteration. |
| RELPARMAX | real | > 0.0 | Parameter change limit for relative-limited parameters. |
| FACPARMAX | real | > 1.0 | Parameter change limit for factor-limited parameters. |
| FACORIG | real | > 0.0<br>< 1.0 | Used to modify calculation of relative parameter change as parameter value approaches zero. |
| PHIREDSWH | real | > 0.0<br>< 1.0 | Relative objective function improvement between iterations below which switch is made to 3 point derivatives calculation. |
| NOPTMAX | integer | ≥ -1 | Maximum number of optimization iterations. |
| PHIREDSTP | real | > 0<br>< 0.1 | Termination criterion: relative objective function change threshold. |
| NPHISTP | integer | > 0 | Number of optimization iterations over which PHIREDSTP applies. |
| NPHINORED | integer | > 0 | Termination criterion: number of optimization iterations since objective function decreased. |
| RELPARSTP | real | > 0.0<br>< 0.1 | Termination criterion: relative parameter change threshold. |
| NRELPAR | integer | > 0 | Number of optimization iterations over which RELPARSTP applies. |
| ICOV | integer | 0 or 1 | If 1, write parameter covariance matrix to matrix file during every optimization iteration. |
| ICOR | integer | 0 or 1 | If 1, write parameter correlation coefficient matrix to matrix file during every optimization iteration. |
| IEIG | integer | 0 or 1 | If 1, write eigenvalues/eigenvectors of parameter covariance matrix to matrix file during every optimization iteration. |

**Table 7. Variables cited in the "control data" section of the PEST control file.**

The NOPTMAX variable sets the maximum number of optimization iterations that PEST will be allowed to carry out. If this is set to 0, PEST will run the model only once; it will then compute the objective function, write this to the screen and to its run record file, and then cease execution. If NOPTMAX is set to -1, PEST will calculate the objective function, Jacobian matrix and various statistical entities such as the parameter covariance matrix and quantities derived from it. It will then cease execution after recording these to its run record file.

| Variable name | Type | Values | Role of variable |
|---|---|---|---|
| PARGPNME | character | *user's choice* (≤ 12 characters) | Name of parameter group. |
| INCTYP | character | "relative" "absolute" | Increment type for derivatives calculation. |
| DERINC | real | > 0.0 | Value of parameter increment. |
| DERINCLB | real | ≥ 0.0 | Absolute lower limit of parameter increment. |
| FORCEN | character | "always_2" "always_3" "switch" | Indicates whether derivatives will be calculated using 2 or 3 points, or whether PEST will start at 2 and switch to 3-point derivatives calculation as required. |
| DERINCMUL | real | > 0.0 | Multiplier for increment if 3-point derivatives calculation is employed. |
| DERMTHD | character | "parabolic" "outside_pts" "best_fit" | Indicates whether derivatives are calculated using parabolic, outside-points or best-fit method if 3-point derivatives calculation is employed. |

**Table 8. Variables cited in the "parameter groups" section of the PEST control file.**

| Variable name | Type | Values | Role of variable |
|---|---|---|---|
| PARNME | character | *user's choice* (≤ 12 characters) | Name of parameter. |
| PARTRANS | character | "none" "log" "fixed" "tied" | Denotes whether parameter is adjustable, tied, fixed or log-transformed. |
| PARCHGLIM | character | "relative" "factor" | Type of change limit - relative or factor. |
| PARVAL1 | real | *problem-specific* | Initial parameter value. |
| PARLBND | real | *problem-specific* | Parameter lower bound. |
| PARUBND | real | *problem-specific* | Parameter upper bound. |
| PARGP | character | *user's choice* (≤ 12 characters) | Group to which parameter belongs. |
| SCALE | real | *problem-specific* | Multiplier for parameter value before being written to model input file. |
| OFFSET | real | *problem-specific* | Offset for parameter value before being written to model input file. |
| DERCOM | integer | > 1 | Command number used to run the model when calculating derivatives of this parameter using finite differences. |

**Table 9. Variables cited in the "parameter data" section of the PEST control file.**

If any parameter is tied to another during the parameter estimation process, the "parameter data" section of the PEST control file must also contain a list of tied parameters together with the parameters to which they are tied.

| **Variable name** | **Type** | **Values** | **Role of variable** |
|---|---|---|---|
| OBGNME | character | *user's choice* (≤ 12 characters) | Name of observation group. |
| COVFLE | character | *user's choice* | Optional name of observation covariance matrix file if a covariance matrix is supplied for this observation group in lieu of observation weights. |

**Table 10. Variables cited in the "observation groups" section of the PEST control file.**

| **Variable name** | **Type** | **Values** | **Role of variable** |
|---|---|---|---|
| OBSNME | character | *user's choice* (≤ 12 characters) | Name of observation. |
| OBSVAL | real | *problem-specific* | Measured value. |
| WEIGHT | real | ≥ 0.0 | Observation weight used in parameter estimation process. |
| OBGNME | character | *user's choice* (≤ 12 characters) | Group to which observation belongs. |

**Table 11. Variables cited in the "observation data" section of the PEST control file.**

| **Variable name** | **Type** | **Values** | **Role of variable** |
|---|---|---|---|
| model command line | character | *problem-specific* | Command issued by PEST to run the model. |

**Table 12. Variables cited in the "model command line" section of the PEST control file.**

| **Variable name** | **Type** | **Values** | **Role of variable** |
|---|---|---|---|
| model command line | character | *problem-specific* | Command issued by PEST to run the model when it is used to calculate parameter derivatives. |
| EXTDERFLE | character | *problem-specific* | The name of the file in which the model records parameter derivatives. |

**Table 13. Variables cited in the optional "derivatives command line" section of the PEST control file.**

A PEST control file will require a "derivatives command line" section only if the JACFILE variable in the "control data" section of the PEST control file is set to 1.

| Variable name | Type | Values | Role of variable |
|---|---|---|---|
| TEMPFLE | character | *user's choice* | Name of template file. |
| INFLE | character | *problem-specific* | Name of model input file corresponding to template file. |
| INSFLE | character | *user's choice* | Name of instruction file. |
| OUTFLE | character | *problem-specific* | Name of model output file corresponding to instruction file. |

**Table 14. Variables cited in the "model input/output" section of the PEST control file.**

All files cited in Table 14 will be assumed to reside in the current working directory; alternatively, inclusion of a path in their name will allow them to reside in a directory of the user's choice.

| Variable name | Type | Values | Role of variable |
|---|---|---|---|
| PILBL | character | *user's choice* (≤ 12 characters) | Name of prior information item. |
| PIFAC | real | *problem-specific* | Factor appearing in prior information equation. |
| PARNME | character | *user's choice* (≤ 12 characters) | Name of parameter appearing in prior information equation. |
| PIVAL | real | *problem-specific* | Value of prior information equation. |
| WEIGHT | real | ≥ 0.0 | Weight assigned to prior information equation. |
| OBGNME | character | *user's choice* (≤ 12 characters) | Observation group to which prior information equation belongs. |

**Table 15. Variables cited in the "prior information" section of the PEST control file.**

| Variable name | Type | Values | Role of variable |
|---|---|---|---|
| NPREDMAXMIN | integer | 1 or -1 | Indicates whether to maximize or minimize prediction. |
| PD0 | real | $> 0.0$ | $\Phi_0$ of equation (2), i.e. upper allowable value for objective function. |
| PD1 | real | > PD0 | Acceptable upper limit for objective function. |
| PD2 | real | > PD1 | Objective function value below which PEST tries to maximize/minimize prediction instead of minimizing objective function. |
| ABSPREDLAM | real | $\geq 0.0$ | Absolute threshold of prediction improvement below which to cease trying new Marquardt lambdas. |
| RELPREDLAM | real | $\geq 0.0$ | Relative threshold of prediction improvement below which to cease trying new Marquardt lambdas. |
| INITSCHFAC | real | $> 0.0$ | Initial factor by which to multiply upgrade vector in line search for maximum/minimum prediction. |
| MULSCHFAC | real | $> 1.0$ | Factor by which upgrade vector is multiplied in line search. |
| NSEARCH | integer | $\geq 1$ | Maximum number of model runs to undertake in line search for prediction maximum/minimum. |
| ABSPREDSWH | real | $\geq 0.0$ | Absolute limit of prediction improvement between optimization iterations below which to switch to 3-pt derivatives calculation. |
| RELPREDSWH | real | $\geq 0.0$<br>$< 0.8$ | Relative limit of prediction improvement between optimization iterations below which to switch to 3-pt derivatives calculation. |
| NPREDNORED | integer | $\geq 1$ | Termination criterion: maximum number of optimization iterations since prediction has been raised or lowered. |
| ABSPREDSTP | real | $\geq 0.0$ | Threshold value of absolute prediction improvement below which optimization process will cease. |
| RELPREDSTP | real | $\geq 0.0$<br>$< 0.2$ | Threshold value of relative prediction improvement below which optimization process will cease. |
| NPREDSTP | integer | $\geq 1$ | Number of iterations over which the previous two variables apply. |

**Table 16. Variables cited in the "predictive analysis" section of the PEST control file.**

| Variable name | Type | | Role of variable |
|---|---|---|---|
| PHIMLIM | real | > 0.0 | Upper limit of acceptability of measurement objective function. |
| PHIMACCEPT | real | > PHIMLIM | Acceptable upper limit for measurement objective function if PHIMLIM criterion cannot be met. |
| FRACPHIM | real | < 1.0 | Factor by which a temporary value for PHIMLIM is calculated from the current value of the measurement objective function. |
| WFINIT | real | > 0.0 | Initial weight factor for regularization observations. |
| WFMIN | real | ≤ WFINIT | Minimum allowable weight factor. |
| WFMAX | real | ≥ WFINIT | Maximum allowable weight factor. |
| WFFAC | real | > 1.0 | Factor by which weight factor is multiplied to begin iterative weight factor solution process. |
| WFTOL | real | > 0.0<br>< 0.2 | Convergence criterion in iterative weight factor solution process. |

**Table 17. Variables cited in the "regularization" section of the PEST control file.**

When Parallel PEST is used, one extra PEST input file will need to be prepared, viz. a "run management file". The structure of this file is shown in Figure 14. An explanation of the variables required by this file is provided in Table 18.

```
prf
NSLAVE IFLETYP WAIT PARLAM
SLAVNAME SLAVDIR
(once for each slave)
(RUNTIME(I), I=1,NSLAVE)
Any lines after this point are required only if IFLETYP is nonzero; the
following group of lines is to be repeated once for each slave.
INFLE(1)
INFLE(2)
(to NTPFLE lines, where NTPFLE is the number of template files)
OUTFLE(1)
OUTFLE(2)
(to NINSFLE lines, where NINSFLE is the number of instruction files)
```

**Figure 14. Structure of the Parallel PEST run management file.**

| Variable name | Type | | Role of variable |
|---|---|---|---|
| NSLAVE | integer | ≥ 1 | Number of slaves. |
| IFLETYP | integer | 0 or 1 | Whether to use long or short version of run management file. |
| WAIT | real | > 0.0 | Length of pause in seconds between writing a file and signalling its presence. |
| PARLAM | integer | 0 or 1 | Set to 1 if partial parallelization of the Marquardt Lambda search is required. |
| SLAVNAME | character | *user's choice* | Name of slave. |
| SLAVDIR | character | *user's choice* | Working directory of slave. |
| RUNTIME | real | > 0.0 | Estimated model run time in seconds on slave's machine. |
| INFLE | character | *problem-specific* | Name of model input file on slave's machine. |
| OUTFLE | character | *problem-specific* | Name of model output file on slave's machine. |

**Table 18. Variables cited in the PEST run management file.**

## 4.2 PEST Output Files

### 4.2.1 General

As was discussed above, PEST will produce a number of binary files that will endow it with restart capabilities. These files will be read by PEST upon re-commencement of execution. Only one of these file will be read by a program other than PEST, this being the Jacobian matrix file. Program JACWRIT, one of the utilities supplied with PEST will re-write the Jacobian matrix file in ASCII format if the user desires. No other external program will read these binary files.

PEST will also produce a number of ASCII files containing information that can be read by the user and any post-processing software that the user may write. These files are now discussed.

### 4.2.2 Run Record File

The run record file will echo data supplied to PEST through its various input files. Once it has read all of its input files, PEST will continue to record information to the run record file as it progresses through the optimization process. When the process is complete, a comprehensive summary of the outcomes of the process will be available from this file.

Information recorded on the run record file during the course of the optimization process will include the following:-

- current values of all parameters,
- current value of the Marquardt lambda,
- current value of the objective function,
- contribution made to the objective function by different observation groups,
- maximum factor or relative change undergone by any parameter,
- current value of model prediction (predictive analysis mode only),
- current value of measurement and regularization objective functions (regularization mode only),
- current value of the regularization weight factor (regularization mode only).

Information written to the run record file upon completion of the optimization process will include the following:-

- optimized values of all parameters,
- 95% confidence intervals of all parameters based on linearity assumption employed in latest Jacobian matrix calculation,
- parameter covariance matrix,
- parameter correlation coefficients,
- eigenvectors and eigenvalues of parameter covariance matrix,
- maximum/minimum value of key model prediction (predictive analysis mode only),
- objective function and contribution made to the objective function by all observation groups,
- measurement and regularization objective functions (regularization mode only),
- observations, their model-generated counterparts, and residuals,
- prior information residuals,
- maximum, minimum, mean, median and standard deviation of residuals for each observation group and for residuals taken as a whole,
- correlation coefficient (calculated using the method of Cooley and Naff, 1990).

An example of a run record file, in which the format of the various items of information listed on that file is exemplified, is provided in Figure 15.

```
PEST RUN RECORD: CASE  manual


Case dimensions:-

   Number of parameters              :    5
   Number of adjustable parameters   :    3
   Number of parameter groups        :    2
   Number of observations            :   19
   Number of prior estimates         :    2


Model command line:-

   ves
```

```
Jacobian command line:-

   na


Model interface files:-

   Templates:
      ves.tpl
   for model input files:
      ves.inp

   (Parameter values written using single precision protocol.)
   (Decimal point always included.)

   Instruction files:
      ves.ins
   for reading model output files:
      ves.out


PEST-to-model message file:-

   pest.mmf

Derivatives calculation:-

Param Increment   Increment   Increment   Forward or    Multiplier  Method
group type                    low bound   central       (central)   (central)
 ro   relative    1.0000E-03  1.0000E-05   switch         2.000     parabolic
 h    relative    1.0000E-03  1.0000E-05   switch         2.000     parabolic


Parameter definitions:-

Name  Trans-        Change     Initial       Lower          Upper        Group
      formation     limit      value         bound          bound
ro1   fixed          na      0.500000         na             na          none
ro2   log           factor    5.00000      0.100000       10.0000        ro
ro3   tied to ro2    na      0.500000         na             na          ro
h1    none          factor    2.00000      5.000000E-02   100.000        h
h2    log           factor    5.00000      5.000000E-02   100.000        h

Name      Group         Scale          Offset      Model command number
ro1       none          1.00000        0.000000     1
ro2       ro            1.00000        0.000000     1
ro3       ro            1.00000        0.000000     1
h1        h             1.00000        0.000000     1
h2        h             1.00000        0.000000     1


Prior information:-

Prior info    Factor              Parameter           Prior         Weight
name                                                  information
  pi1         1.00000      *         h1        =      2.00000        3.000
  pi2         1.00000      *      log[ro2]     +
              1.00000      *      log[h2]      =      2.60260        2.000


Prior Info Name    Observation Group
  pi1                 group_4
  pi2                 group_4


Observations:-

Observation name     Observation        Weight    Group
     ar1              1.21038           1.000     group_1
     ar2              1.51208           1.000     group_1
     ar3              2.07204           1.000     group_1
     ar4              2.94056           1.000     group_1
     ar5              4.15787           1.000     group_1
     ar6              5.77620           1.000     group_1
     ar7              7.78940           1.000     group_2
     ar8              9.99743           1.000     group_2
     ar9              11.8307           1.000     group_2
     ar10             12.3194           1.000     group_2
     ar11            10.6003          1.000    group_2
     ar12            7.00419          1.000    group_2
     ar13            3.44391          1.000    group_2
     ar14            1.58279          1.000    group_2
     ar15            1.10380          1.000    group_3
     ar16            1.03086          1.000    group_3
     ar17            1.01318          1.000    group_3
     ar18            1.00593          1.000    group_3
     ar19            1.00272          1.000    group_3


Inversion control settings:-

   Initial lambda                                               :  5.0000
   Lambda adjustment factor                                     :  2.0000
   Sufficient new/old phi ratio per iteration                   : 0.40000
   Limiting relative phi reduction between lambdas              : 3.00000E-02
   Maximum trial lambdas per iteration                          :  10

   Maximum  factor  parameter change (factor-limited changes)   :  3.0000
   Maximum relative parameter change (relative-limited changes) :   na
   Fraction of initial parameter values used in computing
   change limit for near-zero parameters                        : 1.00000E-03

   Relative phi reduction below which to begin use of
   central derivatives                                          : 0.10000

   Relative phi reduction indicating convergence                : 0.10000E-01
   Number of phi values required within this range              :   3
   Maximum number of consecutive failures to lower phi          :   3
   Maximum relative parameter change indicating convergence     : 0.10000E-01
   Number of consecutive iterations with minimal param change   :   3
   Maximum number of optimization iterations                    :  30


                                 OPTIMISATION RECORD


INITIAL CONDITIONS:
Sum of squared weighted residuals (ie phi) =  523.8
Contribution to phi from observation group "group_1" = 127.3
Contribution to phi from observation group "group_2" = 117.0
Contribution to phi from observation group "group_3" = 185.2
Contribution to phi from prior information           = 94.28


      Current parameter values
        ro1        0.500000
        ro2         5.00000
        ro3        0.500000
        h1          2.00000
        h2          5.00000


OPTIMISATION ITERATION NO.        :    1
   Model calls so far             :    1
   Starting phi for this iteration:  523.8
   Contribution to phi from observation group "group_1": 127.3
   Contribution to phi from observation group "group_2": 117.0
   Contribution to phi from observation group "group_3": 185.2
   Contribution to phi from prior information          : 94.28

   Lambda =  5.000      ----->
      phi =  361.4      (  0.69 of starting phi)

   Lambda =  2.500      ----->
      phi =  357.3      (  0.68 of starting phi)

   No more lambdas: relative phi reduction between lambdas less than 0.0300
   Lowest phi this iteration:  357.3

      Current parameter values                Previous parameter values
        ro1        0.500000                     ro1        0.500000
        ro2         10.0000                     ro2         5.00000
        ro3         1.00000                     ro3        0.500000
      h1          1.94781                        h1          2.00000
      h2          10.4413                        h2          5.00000
   Maximum   factor parameter change:  2.088  [h2]
   Maximum relative parameter change:  1.088  [h2]


OPTIMISATION ITERATION NO.        :    2
   Model calls so far             :    6
   Starting phi for this iteration:  357.3
   Contribution to phi from observation group "group_1": 77.92
   Contribution to phi from observation group "group_2": 103.8
   Contribution to phi from observation group "group_3": 121.3
   Contribution to phi from prior information          : 54.28

   Lambda =  1.250     ----->
      parameter "ro2" frozen: gradient and update vectors out of bounds
      phi =  252.0      (  0.71 of starting phi)

   Lambda = 0.6250     ----->
      phi =  243.6      (  0.68 of starting phi)

   Lambda = 0.3125     ----->
      phi =  235.9      (  0.66 of starting phi)

   Lambda = 0.1563     ----->
      phi =  230.1      (  0.64 of starting phi)

   No more lambdas: relative phi reduction between lambdas less than 0.0300
   Lowest phi this iteration:  230.1

      Current parameter values                 Previous parameter values
        ro1        0.500000                      ro1        0.500000
        ro2         10.0000                      ro2         10.0000
        ro3         1.00000                      ro3         1.00000
        h1          1.41629                      h1          1.94781
        h2          31.3239                      h2          10.4413
   Maximum   factor parameter change:  3.000  [h2]
   Maximum relative parameter change:  2.000  [h2]


OPTIMISATION ITERATION NO.        :    3
   Model calls so far             :   13
   Starting phi for this iteration:  230.1
   Contribution to phi from observation group "group_1": 29.54
   Contribution to phi from observation group "group_2": 84.81
   Contribution to phi from observation group "group_3": 91.57
   Contribution to phi from prior information          : 24.17

   All frozen parameters freed

   Lambda = 7.8125E-02 ----->
      parameter "ro2" frozen: gradient and update vectors out of bounds
      phi =  89.49      (  0.39 of starting phi)

   No more lambdas: phi is now less than 0.4000 of starting phi
   Lowest phi this iteration:  89.49

      Current parameter values                 Previous parameter values
        ro1        0.500000                      ro1        0.500000
        ro2         10.0000                      ro2         10.0000
        ro3         1.00000                      ro3         1.00000
        h1         0.472096                      h1          1.41629
        h2          34.3039                      h2          31.3239
   Maximum   factor parameter change:  3.000  [h1]
   Maximum relative parameter change: 0.6667  [h1]


OPTIMISATION ITERATION NO.        :    4
   Model calls so far             :   17
   Starting phi for this iteration:  89.49
   Contribution to phi from observation group "group_1": 9.345
   Contribution to phi from observation group "group_2": 34.88
   Contribution to phi from observation group "group_3": 21.57
   Contribution to phi from prior information          : 23.69

   All frozen parameters freed

   Lambda = 3.9063E-02 ----->
```

```
      parameter "ro2" frozen: gradient and update vectors out of bounds
      phi =  79.20      (  0.89 of starting phi)

   Lambda = 1.9531E-02 ----->
      phi =  79.19      (  0.88 of starting phi)

   No more lambdas: relative phi reduction between lambdas less than 0.0300
   Lowest phi this iteration:  79.19

      Current parameter values                 Previous parameter values
        ro1        0.500000                      ro1        0.500000
        ro2         10.0000                      ro2         10.0000
        ro3         1.00000                      ro3         1.00000
        h1         0.157365                      h1         0.472096
        h2          44.2189                      h2          34.3039
   Maximum   factor parameter change:  3.000  [h1]
   Maximum relative parameter change: 0.6667  [h1]


OPTIMISATION ITERATION NO.        :    5
   Model calls so far             :   22
   Starting phi for this iteration:  79.19
   Contribution to phi from observation group "group_1": 6.920
   Contribution to phi from observation group "group_2": 22.45
   Contribution to phi from observation group "group_3": 14.88
   Contribution to phi from prior information          : 34.94

   All frozen parameters freed

   Lambda = 9.7656E-03 ----->
      parameter "ro2" frozen: gradient and update vectors out of bounds
      phi =  64.09      (  0.81 of starting phi)

   Lambda = 4.8828E-03 ----->
      phi =  64.09      (  0.81 of starting phi)

   Lambda = 1.9531E-02 ----->
      phi =  64.09      (  0.81 of starting phi)

   No more lambdas: relative phi reduction between lambdas less than 0.0300
   Lowest phi this iteration:  64.09

      Current parameter values                 Previous parameter values
        ro1        0.500000                      ro1        0.500000
        ro2         10.0000                      ro2         10.0000
        ro3         1.00000                      ro3         1.00000
        h1         0.238277                      h1         0.157365
        h2          42.4176                      h2          44.2189
   Maximum   factor parameter change:  1.514  [h1]
   Maximum relative parameter change: 0.5142  [h1]


OPTIMISATION ITERATION NO.        :    6
   Model calls so far             :   28
   Starting phi for this iteration:  64.09
   Contribution to phi from observation group "group_1": 6.740
   Contribution to phi from observation group "group_2": 18.98
   Contribution to phi from observation group "group_3": 10.53
   Contribution to phi from prior information          : 27.84

   All frozen parameters freed

   Lambda = 1.9531E-02 ----->
      parameter "ro2" frozen: gradient and update vectors out of bounds
      phi =  63.61      (  0.99 of starting phi)

   Lambda = 9.7656E-03 ----->
      phi =  63.61      (  0.99 of starting phi)

   No more lambdas: relative phi reduction between lambdas less than 0.0300
   Lowest phi this iteration:  63.61
   Relative phi reduction between optimization iterations less than 0.1000
   Switch to central derivatives calculation

      Current parameter values                 Previous parameter values
        ro1        0.500000                      ro1        0.500000
        ro2         10.0000                      ro2         10.0000
        ro3         1.00000                      ro3         1.00000
        h1         0.265320                      h1         0.238277
        h2          42.2249                      h2          42.4176
   Maximum   factor parameter change:  1.113  [h1]
   Maximum relative parameter change: 0.1135  ]h1]


OPTIMISATION ITERATION NO.         :    7
   Model calls so far              :   33
   Starting phi for this iteration:  63.61
   Contribution to phi from observation group "group_1": 3.679
   Contribution to phi from observation group "group_2": 32.58
   Contribution to phi from observation group "group_3": 0.111
   Contribution to phi from prior information          : 27.24

   All frozen parameters freed

   Lambda = 4.8828E-03 ----->
      parameter "ro2" frozen: gradient and update vectors out of bounds
      phi =  63.59      (  1.00 of starting phi)

   Lambda = 2.4414E-03 ----->
      phi =  63.59      (  1.00 of starting phi)

   Lambda = 9.7656E-03 ----->
      phi =  63.59      (  1.00 of starting phi)

   No more lambdas: relative phi reduction between lambdas less than 0.0300
   Lowest phi this iteration:  63.59

      Current parameter values                 Previous parameter values
        ro1        0.500000                      ro1        0.500000
        ro2         10.0000                      ro2         10.0000
        ro3         1.00000                      ro3         1.00000
        h1         0.261177                      h1         0.265320
        h2          42.2006                      h2          42.2249
   Maximum   factor parameter change:  1.016      [h1]
   Maximum relative parameter change: 1.5615E-02 [h1]

   Optimization complete: the  3 lowest phi's are within a relative distance
                          of eachother of 1.000E-02
   Total model calls:   42


                             OPTIMISATION RESULTS


Adjustable parameters ----->

Parameter      Estimated          95% percent confidence limits
               value              lower limit       upper limit
 ro2           10.0000           0.665815          150.192
 h1           0.261177          -1.00256           1.52491
 h2            42.2006           0.467914          3806.02

Note: confidence limits provide only an indication of parameter uncertainty.
      They rely on a linearity assumption which  may not extend as far in
      parameter space as the confidence limits themselves - see PEST manual.


Tied parameters ----->

Parameter      Estimated value
 ro3             1.00000


Fixed parameters ----->

Parameter      Fixed value
 ro1            0.500000


Observations ----->

Observation    Measured        Calculated       Residual         Weight    Group
               value           value
 ar1           1.21038         1.64016        -0.429780         1.000    group_1
 ar2           1.51208         2.25542        -0.743340         1.000    group_1
 ar3           2.07204         3.03643        -0.964390         1.000    group_1
 ar4          2.94056        3.97943       -1.03887        1.000     group_1
 ar5          4.15787        5.04850      -0.890630        1.000     group_1
 ar6          5.77620        6.16891      -0.392710        1.000     group_1
 ar7          7.78940        7.23394       0.555460        1.000     group_2
 ar8          9.99743        8.12489        1.87254        1.000     group_2
 ar9          11.8307        8.72551        3.10519        1.000     group_2
 ar10         12.3194        8.89590        3.42350        1.000     group_2
 ar11         10.6003        8.40251        2.19779        1.000     group_2
 ar12         7.00419        6.96319      4.100000E-02     1.000     group_2
 ar13         3.44391        4.70412       -1.26021        1.000     group_2
 ar14         1.58279        2.56707      -0.984280        1.000     group_2
 ar15         1.10380        1.42910      -0.325300        1.000     group_3
 ar16         1.03086        1.10197     -7.111000E-02     1.000     group_3
 ar17         1.01318        1.03488     -2.170000E-02     1.000     group_3
 ar18         1.00593        1.01498     -9.050000E-03     1.000     group_3
 ar19         1.00272        1.00674     -4.020000E-03     1.000     group_3


Prior information ----->

Prior          Provided       Calculated       Residual        Weight
information    value          value
 pi1           2.00000       0.261177         1.73882         3.000
 pi2           2.60260        2.62532      -2.271874E-02      2.000

See file TEMP3.RES for more details of residuals in graph-ready format.
See file TEMP3.SEO for composite observation sensitivities.

Objective Function ----->

  Sum of squared weighted residuals (ie phi)                 =   63.59
  Contribution to phi from observation group "group_1"       =   3.686
  Contribution to phi from observation group "group_2"       =   32.58
  Contribution to phi from observation group "group_3"       =  0.1115
  Contribution to phi from prior information                 =   27.21


Correlation Coefficient ----->
  Correlation coefficient                                    =  0.9086


Analysis of residuals ----->

  All residuals:-
     Number of residuals with non-zero weight                =    21
     Mean value of non-zero weighted residuals               = -0.4399
     Maximum weighted residual [observation "ar13"]          =   1.260
     Minimum weighted residual [observation "pi1"]           =  -5.216
     Standard variance of weighted residuals                 =   3.533
     Standard error of weighted residuals                    =   1.880

     Note: the above variance was obtained by dividing the objective
     function by the number of system degrees of freedom (ie. number of
     observations with non-zero weight plus number of prior information
     articles with non-zero weight minus the number of adjustable parameters.)
     If the degrees of freedom is negative the divisor becomes
     the number of observations with non-zero weight plus the number of
     prior information items with non-zero weight.

  Residuals for observation group "group_1":-
     Number of residuals with non-zero weight                =     6
     Mean value of non-zero weighted residuals               =  0.7424
     Maximum weighted residual [observation "ar4"]           =   1.038
     Minimum weighted residual [observation "ar6"]           =  0.3916
     "Variance" of weighted residuals                        =  0.6144
     "Standard error" of weighted residuals                  =  0.7838

     Note: the above "variance" was obtained by dividing the sum of squared
     residuals by the number of items with non-zero weight.

  Residuals for observation group "group_2":-
     Number of residuals with non-zero weight                =     8
     Mean value of non-zero weighted residuals               =  -1.119
     Maximum weighted residual [observation "ar13"]          =   1.260
     Minimum weighted residual [observation "ar10"]          =  -3.424
     "Variance" of weighted residuals                        =   4.072
     "Standard error" of weighted residuals                  =   2.018
```

```
    Note: the above "variance" was obtained by dividing the sum of squared
    residuals by the number of items with non-zero weight.

 Residuals for observation group "group_3":-
    Number of residuals with non-zero weight                =     5
    Mean value of non-zero weighted residuals               =  8.6256E-02
    Maximum weighted residual [observation "ar15"]          =  0.3254
    Minimum weighted residual [observation "ar19"]          =  4.0200E-03
    "Variance" of weighted residuals                        =  2.2300E-02
    "Standard error" of weighted residuals                  =  0.1493

    Note: the above "variance" was obtained by dividing the sum of squared
    residuals by the number of items with non-zero weight.

 Prior information residuals:-
    Number of residuals with non-zero weight                =     2
    Mean value of non-zero weighted residuals               =  -2.585
    Maximum weighted residual [observation "pi2"]           =  4.5451E-02
    Minimum weighted residual [observation "pi1"]           =  -5.216
    "Variance" of weighted residuals                        =   13.61
    "Standard error" of weighted residuals                  =   3.689

    Note: the above "variance" was obtained by dividing the sum of squared
    residuals by the number of items with non-zero weight.


Covariance Matrix ----->

 0.3136       4.8700E-03  -0.4563
 4.8700E-03   0.3618       1.3340E-02
-0.4563       1.3340E-02   0.8660

Correlation Coefficient Matrix ----->

  1.000       1.4457E-02  -0.8756
 1.4457E-02    1.000       2.3832E-02
-0.8756       2.3832E-02    1.000

Normalized eigenvectors of covariance matrix ----->

-0.8704      -3.6691E-02  -0.4909
 3.5287E-02  -0.9993       1.2121E-02
-0.4910      -6.7718E-03   0.8711

Eigenvalues ----->

 5.6045E-02   0.3621        1.123
```

**Figure 15. Example of a PEST run record file.**

### 4.2.3 Parameter Value File

At the end of every optimization iteration PEST will record the best parameter values that it has calculated so far to a "parameter value file". The definition of "best" will depend on its current mode of operation. The format of the parameter value file will be such that it can be used by the PEST utility program, TEMPCHEK for the writing of model input files. Hence, through the use of TEMPCHEK as a postprocessor for PEST, model input files can be written using optimized parameter values after the optimization process is complete. The parameter value file will also be useable by the utility program PESTGEN in building a new PEST control file, and by the utility program PARREP in modifying an existing PEST control file to contain optimized parameter values calculated by PEST.

An example of a parameter value file is provided in Figure 16. The first line of this file will contain values of the PEST control variables PRECIS and DPOINT which determine the precision and decimal point protocol with which numbers are to be written to model input files. Then follows a line for each parameter. Each line contains a parameter name, followed by the value of the parameter, then the SCALE and OFFSET pertaining to the parameter. Numbers are written in free field format on each line, and should be space or comma-delimited.

```
single point
   ro1      1.000000          1.000000          0.0000000
   ro2      40.00090          1.000000          0.0000000
   ro3      1.000000          1.000000          0.0000000
    h1      1.000003          1.000000          0.0000000
    h2      9.999799          1.000000          0.0000000
```

**Figure 16. A parameter value file.**

### 4.2.4 Parameter Sensitivity File

During each optimization iteration, just after it has calculated the Jacobian matrix, PEST will calculate the "composite parameter sensitivity" of all parameters and record these to a special "parameter sensitivity file". Composite sensitivity is defined by the equation:-

$$s_i = (\mathbf{J}^t\mathbf{QJ})_{ii}^{1/2} / m \qquad (4)$$

where:-

$s_i$ is the composite sensitivity of the $i^{th}$ parameter,

$\mathbf{J}$ is the Jacobian matrix,

$\mathbf{Q}$ is the cofactor matrix (see section 3.4), and

$m$ is the number of observations and prior information equations involved in the parameter estimation process.

The composite sensitivity of a parameter is one measure of the extent to which its value can be inferred on the basis of the observation data set available for use in the calibration process.

The parameter sensitivity file will be updated from iteration to iteration. Thus at the end of the optimization process it will record the "sensitivity history" of all parameters.

Part of a parameter sensitivity file is illustrated in Figure 17. During each optimization iteration PEST will list all parameters sequentially, with one line devoted to each parameter. Each such line will contain the name of the parameter, the group to which the parameter belongs, the current value of the parameter, the composite sensitivity of that parameter calculated according to equation (4), and the relative composite sensitivity of the parameter. The relative composite sensitivity of a parameter will be obtained by multiplying its composite sensitivity by the magnitude of the value of the parameter. It is thus a measure of the composite changes in model outputs that are incurred by a fractional change in the value of the parameter. Composite sensitivities recorded in the parameter sensitivity file will be sensitivities "as PEST sees them". Thus if a parameter is log-transformed, sensitivity will be expressed with respect to the log of that parameter. See the PEST User's manual for further details.

```
          PARAMETER SENSITIVITIES: CASE VESRA


  OPTIMISATION ITERATION NO.  1 ----->
  Parameter name     Group          Current value     Sensitivity       Rel. Sensitivity
    ro1              ro               1.00000          2.831977E-02        0.00000
    ro2              ro               1.00000          7.000557E-02        0.00000
    ro3              ro               1.00000          6.521583E-02        0.00000
    ro4              ro               1.00000          5.928780E-02        0.00000

    ro5              ro               1.00000          5.668304E-02        0.00000
    ro6              ro               1.00000          5.544761E-02        0.00000
    ro7              ro               1.00000          5.489995E-02        0.00000
    ro8              ro               1.00000          5.459381E-02        0.00000
    ro9              ro               1.00000          5.436991E-02        0.00000
    ro10             ro               1.00000          0.119913            0.00000


  OPTIMISATION ITERATION NO.  2 ----->
  Parameter name     Group          Current value     Sensitivity       Rel. Sensitivity
    ro1              ro              0.999500          5.323333E-02      1.157362E-05
    ro2              ro               1.03878          9.246515E-02      1.527886E-03
    ro3              ro               1.24743          9.865513E-02      9.472454E-03
    ro4              ro               1.75286          0.101822          2.481886E-02
    ro5              ro               2.59093          0.107088          4.427621E-02
    ro6              ro               3.00000          0.117925          5.626457E-02
    ro7              ro               2.16466          0.109469          3.671474E-02
    ro8              ro               1.28441          8.904916E-02      9.679949E-03
    ro9              ro              0.982469          7.757826E-02      5.958761E-04
    ro10             ro              0.965695          0.133894          2.029866E-03
```

**Figure 17. Part of a parameter sensitivity file.**

At the end of the parameter estimation process PEST will provide a complete listing of composite parameter sensitivities based on the best sensitivity matrix (i.e. Jacobian matrix) computed during the optimization process. “Best” will be defined in terms of the aim of the optimization process; this may be to minimize the objective function (parameter estimation mode), to maximize/minimize a prediction subject to objective function constraints (predictive analysis mode), or to minimize the regularization component of the objective function subject to constraints imposed on the measurement component of the objective function (regularization mode).

When writing these “completion parameter sensitivities” to the end of the parameter sensitivity file, PEST will list the composite sensitivity and relative composite sensitivity of each parameter with respect to all observation groups, as well as with respect to each individual observation group. The composite parameter sensitivity of each observation group will be evaluated using equation 4 with the summation implied by the matrix equation confined to members of that particular observation group only. The magnitude will then be divided by the number of members of that observation group which have non-zero weights. Figure 18 shows an example of “completion parameter sensitivities” to be recorded at the end of a parameter sensitivity file.

```
                         COMPLETION OF OPTIMISATION PROCESS

Composite sensitivities for observation group "obsgp1" ----->

Number of observations with non-zero weight =      8
Parameter name     Group          Current value    Sensitivity        Rel. Sensitivity
  ro1              ro              0.889162         1.02061            5.207045E-02
  ro2              ro               1.39300         1.34820            0.194075
  ro3              ro               4.47273        0.845800            0.550255
  ro4              ro               16.1338        0.518719            0.626477
  ro5              ro               35.0896        0.522264            0.806991
  ro6              ro               19.6885        0.279802            0.362123
  ro7              ro               4.60888        4.894390E-02        3.247893E-02
  ro8              ro               1.44009        6.702711E-03        1.061632E-03
  ro9              ro              0.923289        9.130165E-04        3.164747E-05

  ro10             ro              0.939085        1.608980E-04        4.391740E-06


Composite sensitivities for observation group "obsgp2" ----->

Number of observations with non-zero weight =    11
Parameter name     Group          Current value    Sensitivity        Rel. Sensitivity
  ro1              ro              0.889162        0.485804            2.478527E-02
  ro2              ro               1.39300        0.637778            9.180881E-02
  ro3              ro               4.47273        0.420981            0.273879
  ro4              ro               16.1338        0.416505            0.503029
  ro5              ro               35.0896         1.20482             1.86166
  ro6              ro               19.6885         1.28907             1.66833
  ro7              ro               4.60888        0.549215            0.364456
  ro8              ro               1.44009        0.274187            4.342806E-02
  ro9              ro              0.923289        0.196916            6.825610E-03
  ro10             ro              0.939085        0.343047            9.363542E-03



Composite sensitivities for all observations/prior info ----->

Number of observations with non-zero weight =    29
Parameter name     Group          Current value    Sensitivity        Rel. Sensitivity
  ro1              ro              0.889162        0.348292            1.776955E-02
  ro2              ro               1.39300        0.452692            6.516547E-02
  ro3              ro               4.47273        0.296685            0.193015
  ro4              ro               16.1338        0.231341            0.279399
  ro5              ro               35.0896        0.487534            0.753326
  ro6              ro               19.6885        0.503110            0.651131
  ro7              ro               4.60888        0.227297            0.150833
  ro8              ro               1.44009        0.137489            2.177661E-02
  ro9              ro              0.923289        0.116886            4.051560E-03
  ro10             ro              0.939085        0.158161            4.317032E-03
```

**Figure 18. An example of parameter sensitivity information recorded at the end of the parameter sensitivity file upon completion of PEST execution.**

### 4.2.5 Observation Sensitivity File

At the end of the optimization process PEST will record the "composite observation sensitivity" of every observation involved in this process to a special "observation sensitivity file". The composite sensitivity of the *j*th observation is defined by the equation:-

$$s_j = \{\mathbf{Q}(\mathbf{J}\mathbf{J}^t)\}_{j,j}{}^{1/2}/n \qquad (5)$$

where symbols are the same as for equation (4) and *n* is the number of adjustable parameters. The composite sensitivity of an observation is a measure of the "strength" of that observation in allowing parameters to be estimated through the optimization process; however its use is limited because it does not take into account the effect of other observations of similar type. Hence a high value for the composite sensitivity of a particular observation does not guarantee that the observation is not redundant.

An example of an observation sensitivity file is provided in Figure 19. One line of data is written for each observation used in the parameter estimation process. Each such line lists, in order, the observation name, the group to which the observation belongs, the measured observation value, the corresponding model-generated value and the composite sensitivity of the observation calculated according to equation (5).

```
Observation   Group          Measured             Modelled             Sensitivity
  ar1           group_1        1.210380             1.639640             0.5221959
  ar2           group_1        1.512080             2.254750             0.6824375
  ar3           group_1        2.072040             3.035590             0.8591846
  ar4           group_1        2.940560             3.978450             1.0338167
  ar5           group_1        4.157870             5.047430             1.1915223
  ar6           group_1        5.776200             6.167830             1.3226952
  ar7           group_2        7.789400             7.232960             1.4450249
  ar8           group_2        9.997430             8.124100             1.5881968
  ar9           group_2        11.83070             8.724950             1.7506757
  ar10          group_2        12.31940             8.895600             1.8875951
```

**Figure 19. Part of an observation sensitivity file.**

### 4.2.6 Residuals File

At the end of the optimization process, PEST will record the following information to a "residuals file".

- field or laboratory measurements used in the calibration process,
- model-generated equivalents to these,
- residuals,
- weighted measurements,
- weighted model-generated counterparts to measurements,
- weighted residuals,
- measurement standard deviations, and
- natural weights.

The "measurement standard deviation" of each observation will be calculated as the inverse of its weight multiplied by the square root of the "reference variance"; the reference variance is equal to the variance of the weighted residuals. "Natural weights", as represented in the residuals file, are the inverse of measurement standard deviations.

Data stored within the residuals file will adhere to a format whereby it is easily imported into a spreadsheet for further post-processing and analysis.

The character width of the residuals file is too great to provide an example in this document to illustrate its formatting; so its formatting will be described. One record will be written for each observation comprising the calibration data set. Each number written to this file (see the above list for what these numbers represent) will be recorded with a field width of 14 characters, separated from its neighbor by two blank characters. The field width of the observation name (which will lead each line) will be 12 characters.

## 4.2.7 Matrix File

If any of the ICOV, ICOR or IEIG variables in the "control data" section of the PEST control file are set to 1, PEST will compute the parameter covariance matrix during each optimization iteration on the basis of current parameter values. It will then write to a "matrix file" one or all (depending on the settings of the above variables) of the following data:

- parameter standard deviations,
- the parameter covariance matrix,
- the parameter correlation coefficient matrix, and
- the eigenvalues and eigenvectors of the parameter covariance matrix.

Figure 20 shows an example of a matrix file written by PEST.

```
PARAMETER STATISTICAL MATRICES: CASE VES7

 OPTIMISATION ITERATION NUMBER 12


 Parameter standard deviations ----->

 Adjustable          Group            Current           Standard
 parameter                            value             deviation
 ro1                 ro               0.999991          6.950057E-06
 ro2                 ro                40.0038          3.427096E-05
 h1                  hhh               1.00001          9.515758E-06
 h2                  hhh               9.99902          3.501148E-05

 Note that if a parameter is log-transformed, the standard deviation in the
 above table refers to the log of the parameter. Note also that the objective
 function used in the calculation of the covariance matrix (and eigenvalues
 of the covariance matrix) was calculated at the end of the previous iteration.


 Parameter covariance matrix ----->

                  ro1          ro2           h1           h2
 ro1          4.8303E-11   6.0912E-11   5.5980E-11  -6.7643E-11
 ro2          6.0912E-11   1.1745E-09   2.3340E-10  -1.1988E-09
 h1           5.5980E-11   2.3340E-10   9.0550E-11  -2.4321E-10
 h2          -6.7643E-11  -1.1988E-09  -2.4321E-10   1.2258E-09


 Parameter correlation coefficient matrix ----->

                  ro1          ro2           h1           h2
 ro1            1.000       0.2557       0.8464      -0.2780
 ro2           0.2557        1.000       0.7157      -0.9991
 h1            0.8464       0.7157        1.000      -0.7300
 h2           -0.2780      -0.9991      -0.7300        1.000


 Normalized eigenvectors of covariance matrix ----->

               Vector_1     Vector_2     Vector_3     Vector_4
 ro1            0.5124       0.4761      -0.7135     -4.0742E-02
 ro2            0.5479      -0.4518       0.1315     -0.6916
```

```
h1            -0.4625      -0.5430      -0.6863      -0.1422
h2             0.4726      -0.5238      -5.0493E-02   0.7070


Eigenvalues ----->

               5.2599E-04   1.0707E-02   8.6137E-02   2.4514E-01
```

**Figure 20. Contents of a matrix file written by PEST.**

The matrix file will be re-written during every optimization iteration as parameters are upgraded through the parameter estimation process.

### 4.2.8 Run Management Record File

Parallel PEST will write a "run management record file" recording the names and details of all slaves used in the parameter estimation process, and the history of communications between PEST and each of its slaves. Each communication will be annotated with the time at which the communication took place. An example of such a file, illustrating its formatting, is provided in Figure 21.

```
PEST RUN MANAGEMENT RECORD FILE: CASE VES2

 SLAVE DETAILS:-

 Slave Name                  PSLAVE Working Directory
 ----------                  ------------------------
 "slave 1"                   ./model1/
 "slave 2"                   ./model2/
 "slave 3"                   ./model3/


 Attempting to communicate with slaves ....

 - slave "slave 2" has been detected.
 - slave "slave 3" has been detected.
 - slave "slave 1" has been detected.


 SLAVE MODEL INPUT AND OUTPUT FILES:-

 Slave "slave 1" ----->

     Model input files on slave "slave 1":-
         ./model1/ves.in1
         ./model1/ves.in2

     Model output files on slave "slave 1":-
         ./model1/ves.ot1
         ./model1/ves.ot2
         ./model1/ves.ot3

     Model command line for slave "slave 1":-
         ves


 Slave "slave 2" ----->

     Model input files on slave "slave 2":-
         ./model2/ves.in1
         ./model2/ves.in2

     Model output files on slave "slave 2":-
         ./model/ves.ot1
         ./model/ves.ot2
         ./model/ves.ot3
```

```
    Model command line for slave "slave 2":-
        ves


 Slave "slave 3" ----->

    Model input files on slave "slave 3":-
        ./model3/ves.in1
        ./model3/ves.in2

    Model output files on slave "slave 3":-
        ./model3/ves.ot1
        ./model3/ves.ot2
        ./model3/ves.ot3

    Model command line for slave "slave 3":-
        ves

AVERAGE WAIT INTERVAL: 50 hundredths of a second.

                            RUN MANAGEMENT RECORD

RUNNING MODEL FOR FIRST TIME  ----->
   21:50:00.19:- slave "slave 1" commencing model run.
   21:55:05.00:- slave "slave 1" finished execution; reading results.


 OPTIMISATION ITERATION NO.  1 ----->

 Calculating Jacobian matrix: running model  5 times .....
   21:55:23.65:- slave "slave 1" commencing model run.
   21:55:33.92:- slave "slave 3" commencing model run.
   21:55:44.20:- slave "slave 2" commencing model run.
   22:00:05.79:- slave "slave 2" finished execution; reading results.
   22:00:17.77:- slave "slave 3" finished execution; reading results.
   22:00:29.47:- slave "slave 2" commencing model run.
   22:00:39.58:- slave "slave 3" commencing model run.
   22:00:50.07:- slave "slave 1" finished execution; reading results.
   22:05:11.83:- slave "slave 2" finished execution; reading results.
   22:05:43.70:- slave "slave 3" finished execution; reading results.

 Testing parameter upgrades .....
   22:06:01.69:- slave "slave 2" commencing model run.
   22:11:16.45:- slave "slave 2" finished execution; reading results.
   22:11:27.49:- slave "slave 2" commencing model run.
   22:16:19.63:- slave "slave 2" finished execution; reading results.
   22:16:35.95:- slave "slave 2" commencing model run.
   22:21:23.48:- slave "slave 2" finished execution; reading results.


 OPTIMISATION ITERATION NO.  2 ----->

 Calculating Jacobian matrix: running model  5 times .....
   22:21:45.07:- slave "slave 2" commencing model run.
   22:21:55.40:- slave "slave 3" commencing model run.
   22:22:25.05:- slave "slave 1" commencing model run.
   22:27:05.83:- slave "slave 2" finished execution; reading results.
   22:27:28.20:- slave "slave 3" finished execution; reading results.
   22:27:39.52:- slave "slave 2" commencing model run.
   22:27:52.63:- slave "slave 3" commencing model run.
   22:28:05.18:- slave "slave 1" finished execution; reading results.
   22:33:06.55:- slave "slave 2" finished execution; reading results.
   22:33:33.03:- slave "slave 3" finished execution; reading results.

 Testing parameter upgrades .....
   22:34:11.46:- slave "slave 2" commencing model run.
   22:39:36.82:- slave "slave 2" finished execution; reading results.
   22:39:48.09:- slave "slave 2" commencing model run.
   22:45:40.28:- slave "slave 2" finished execution; reading results.
   22:45:51.38:- slave "slave 2" commencing model run.
   22:50:43.30:- slave "slave 2" finished execution; reading results.


 OPTIMISATION ITERATION NO.  3 ----->

 Calculating Jacobian matrix: running model 10 times .....
```

```
   22:50:54.46:- slave "slave 2" commencing model run.
   22:51:14.68:- slave "slave 3" commencing model run.
                                  .

                                  .
```

**Figure 21. Part of a Parallel PEST run management record file.**

72

# 5. User Interfaces

## 5.1 Input Data

All input data will be supplied to PEST through its template, instruction and control files (and perhaps one or more observation covariance files) in the manner already discussed. The names of the template and instruction files pertaining to a particular optimization run (and the model input and output files to which they correspond) will be cited in the PEST control file. So, too, will the names of any observation covariance matrix files required for the parameter estimation process carried out by PEST.

## 5.2 Command Line

PEST will be directed to its input control file through its command line. PEST will be run using the command:-

```
pest casefile [/r] [/j]
```

where

| | |
|---|---|
| *casefile* | is the name of the PEST control file pertinent to the current optimization problem, |
| /r | informs PEST that it must re-commence a previous, prematurely-terminated run at the beginning of the optimization iteration during which termination took place, and |
| /j | informs PEST that it must re-commence a previous, prematurely-terminated run at that location within the previous optimization process at which calculation of the Jacobian matrix had just taken place. |

Parallel PEST will be run in identical fashion, except that the "ppest" command will be used instead of the "pest" command. Each of its slaves will be run by typing "pslave" at the screen prompt.

## 5.3 User Intervention

As was discussed in Section 3.11, if he/she deems it necessary, the user will be able to intervene in the optimization process in order to hold troublesome parameters fixed and/or to alter the values of a number of key input control variables. Such intervention will take place by terminating PEST execution, writing or editing a "parameter hold file", and then re-starting PEST using the "/j" switch.

## 5.4 Terminal Display

Throughout the course of its execution PEST will display an abbreviated form of the run record to the terminal screen so that the user can easily monitor the progress of the optimization process. Information displayed on the screen will include the following:-

- optimization iteration number,
- current value of the Marquardt lambda,
- current value of the objective function,
- contribution made to the objective function by different observation groups,
- maximum factor or relative change undergone by any parameter,
- current value of a key model prediction (predictive analysis mode only),
- current value of measurement and regularization objective functions (regularization mode only), and
- current value of the regularization weight factor (regularization mode only).

## 5.5 Error Messages

If it encounters an error in its input data set, PEST will report the error to the screen and to its run record file; it will then terminate execution.

Tables 19 to 23 present a list of PEST error messages. The error messages are grouped according to the categories comprising the table captions. An appropriate user response to each message is also provided.

| **Error message** | **Reason for error** | **User's response** |
|---|---|---|
| All slaves are not alive. Start missing slaves and re-start PEST. | Parallel PEST has not detected the presence of an expected slave. | Check that the slave working directories supplied in the run management file are correct. |
| Cannot close file *filename*. | Parallel PEST is unable to close a model input or output file resident on another machine. | Check that the local area network is operating correctly and that it is not burdened with excessive traffic. |
| Cannot communicate with file *filename*. | Parallel PEST is unable to write or read a message file on another machine. | Check that the slave working directories supplied in the run management file are correct. |
| Cannot delete file *filename*. | Parallel PEST is unable to delete a model output file on another machine. | Check that the local area network is operating correctly and that it is not burdened with excessive traffic. |
| Cannot open file *filename*. | File not present or is erroneously named. | Create file expected by PEST under the name expected by PEST. |
| Cannot open file *filename* for output. | The file is currently in use by another application. | Close the file in the other application. |

| | | |
|---|---|---|
| Cannot open file *filename* to read Jacobian matrix. | The Jacobian matrix file from the previous PEST run has been deleted or is in use by another application. | Close the file in the other application or re-start the current PEST run with "/r" switch. |
| Cannot open file *filename* to read restart data. | One of the binary files that stores the data that PEST needs to re-commence execution after a previous interruption has been deleted. | Recommence the PEST run from the beginning of the optimization process. |
| Cannot open file *filename* to recommence run record. | The run record file is currently in use by another application. | Close the file in the other application. |
| Cannot open model input file *filename* to update parameter values. | The model input file is currently in use by another application. | Close the file in the other application. |
| Cannot open model input template file *filename*. | File is not present or is erroneously named. | Create template file expected by PEST under the name expected by PEST. |
| Cannot open model output file *filename*. | The model output file has not been written by the model, probably because the model did not run correctly. | Attempt to run the model from the command line in order to ascertain the reason for its failure to run. |
| Cannot open instruction file *filename*. | Either the instruction file is incorrectly named in the PEST control file, or it is currently opened by another application. | Alter the name of the instruction file in the PEST control file, or close the file in the other application. |
| Cannot open run management file *filename*. | File is not present or is erroneously named. | Create run management file, or copy existing file to that expected by PEST. |
| Cannot re-open file *filename* to continue PEST run record. | When PEST execution was re-commenced using the "/r" or "/j" switch, it could not re-open the run record file. | Check that the run record file has not been deleted, or is currently held open by another application. |
| Cannot write data to observation sensitivity file. | Observation sensitivity file is currently in use by another application. | Close the file in the other application. |
| Cannot write data to run record file. | The run record file is currently in use by another application. | Close the file in the other application. |
| Cannot write data to parameter sensitivity file. | The parameter sensitivity file is currently in use by another application. | Close the file in the other application. |
| Cannot write data to residuals file. | The residuals file is currently in use by another application. | Close the file in the other application. |

| | | |
|---|---|---|
| Cannot write file *filename* to record restart data. | The binary restart data file is being used by another application (maybe another PEST run). | Close the other application. |
| Cannot write to PEST message file *pest.mmf*. | The operating system will not allow PEST to write to the PEST message file – probably because another application has opened this file. | Close the other application. |
| Cannot write to run management record file. | The run management record file is being used by another application. | Close the file in the other application. |
| Error reading file *filename* to obtain restart data. | The file has been corrupted or the user has altered the PEST control file since the previous run. | Alter the PEST control file to its status before the previous run, or re-commence a new PEST run from the beginning. |
| Error reading line *n* of file *filename*; incorrect data type or data missing. | PEST encountered a character string or blank space where it expected to find a number. | Edit the file and place the correct data item where PEST expects to find it. |
| Error writing to file *filename*. | The file is currently in use by another application. | Close the file in the other application. |
| Unable to write model input file. | The model input file is currently in use by another application. | Close the file in the other application. |
| Unexpected end to file *filename*; data missing. | When reading file *filename*, PEST unexpectedly encountered the end to the file. | Edit the file, adding the data expected by PEST to the end of the file. |
| Unexpected end to file holding restart data. | Either the file is corrupted or the PEST control file has been altered since the previously interrupted run. | Change the PEST control file to its status before the previous run, or re-commence the new PEST run from the beginning. |

**Table 19. File-handling errors.**

| **Error message** | **Reason for error** | **User's response** |
|---|---|---|
| All observations belonging to observation group "regul" have a weight of zero. | When PEST is run in regularization mode, weights assigned to observation group "regul" cannot be altered by a weight factor if they are zero. | Edit the PEST control file, altering the weight assigned to at least one member of observation group "regul" to a non-zero value. |
| Attempt to log transform zero or negative parameter value. | The initial value for a log-transformed parameter supplied in the PEST control is zero or negative. | Untransform the parameter, or keep it positive using appropriate SCALE and OFFSET values. |
| Both PHIMLIM and PHIMACCEPT must be positive in regularization section of PEST control file. | One of PHIMLIM or PHIMACCEPT has been supplied with a zero or negative value. | Edit the PEST control file, supplying positive values for both of these variables. |
| Error in predictive analyzer data set. | An error has been encountered in reading one of the variables from the "predictive analysis" section of the PEST control file. | Edit the PEST control file, rectifying the error. |
| Error reading prior information: line *n* of file *filename*. | Incorrect syntax used in prior information equation. | Edit the PEST control file and rectify the prior information syntax error. |
| FRACPHIM must be less than 1.0 in regularization section of PEST control file. | The value supplied for the FRACPHIM variable in the PEST control file is out of bounds. | Edit the PEST control file, supplying an appropriate value for this variable. |
| Incorrect parameter name or improper syntax in prior information: line *n* of file *filename*. | Incorrect syntax has been used in one of the lines which is used to provide prior information to PEST. | Edit the PEST control file and rectify the prior information syntax error. |
| Log-transformed parameter referenced as untransformed in prior information. Parameter *parname* on line *n* of file *filename*. | A log-transformed parameter must be referenced as such in each item of prior information which cites that parameter. | Either alter the parameter's PARTRANS value to "none" or reference that parameter as log-transformed in the prior information equation. |
| No members of observation group "regul" in "observation data" section of PEST control file. | If PEST is run in regularization mode, one or more observations must belong to the observation group "regul". | Edit the PEST control file, adding observations to the observation group "regul" as appropriate. |
| No observation in the "observation data" section of the PEST control file belongs to the observation group "predict". | If PEST is run in predictive analysis mode, then one observation must be assigned to the observation group "predict". | Edit the PEST control file, assigning the name of the observation that is serving as a prediction to the observation group "predict". |

| | | |
|---|---|---|
| Observation name *obsname* from instruction file not cited in PEST control file. Instruction line follows:- | An instruction has used an observation name which is not cited in the PEST control file. | Add the instruction name, together with its measured value and weight, to the PEST control file. |
| Observation group "predict" not found in "observation groups" section of PEST control file *filename*. | If PEST is run in predictive analysis mode, an observation group named "predict" must be employed. | Add the name of this observation group to the "observation groups" section of the PEST control file. |
| Observation group "regul" not found in "observation groups" section of PEST control file. | If PEST is run in regularization mode, an observation group named "regul" must be employed. | Add the name of this observation group to the "observation groups" section of the PEST control file. |
| Observation *obsname* not referenced in any instruction file. | An observation was encountered in the "observation data" section of the PEST control file which was not cited in any instruction files. | Provide an instruction to read the observation in an instruction file cited in the PEST control file, or remove the instruction from the PEST control file. |
| Only one observation can belong to observation group "predict" in "observation data" section of PEST control file. | PEST is running in predictive analysis mode and more than one observation has been assigned to the observation group "predict". | Edit the PEST control file, ensuring that only one observation belongs to the observation group "predict". |
| Parameter cited in prior information is fixed or tied. Parameter *parname* on line *n* of file *filename*. | All parameters cited in prior information must be adjustable; this rule has been violated. | Edit the PEST control file and rectify the error. |
| Parameter *parname* assigned a SCALE value of zero; line *n* of file *filename*. | A parameter has been assigned a SCALE value of zero; this is an illegal value. | Edit the PEST control file, assigning the parameter a non-zero SCALE value. |
| Parameter *parname* is not a tied parameter. Line *n* of file *filename*. | A parameter, not declared as "tied", has been linked to another parameter. | Alter the PARTRANS value of the offending parameter to "tied". |
| Parameter cannot be tied to fixed or tied parameter: line *n* of file *filename*. | A parameter in the PEST control file is denoted as being tied to a parameter which is itself fixed or tied. | If the parameter is tied to another tied parameter, tie the parameter to the parent of the tied parameter instead. |
| Parameter cannot be tied to itself; line *n* of file *filename*. | A parameter was designated as being tied to itself in the "parameter data" section of the PEST control file. | Either designate the parameter as adjustable, or tie it to another parameter. |
| Parameter group "none" is reserved for tied and fixed parameters. Line *n* of file *filename*. | A parameter was assigned to a parameter group named "none" in the "parameter data" section of the PEST control file. | Assign the parameter to another group. |

| | | |
|---|---|---|
| PESTMODE must be "estimation", "regularization" or "prediction" in line *n* of file *filename*. | An incorrect specification has been set for the PESTMODE variable in the PEST control file. | Edit the PEST control file, altering the value for the PESTMODE variable. |
| Prediction must be final observation when PEST is run in predictive analysis mode and derivatives are supplied externally. | Data pertaining to the prediction in the "observation data" section of the PEST control file must be located at the end of this section. | Edit the "observation data" section of the PEST control file, moving the single line pertaining to the prediction to the end of this section. |
| Third line of file *filename* must be "restart" or "norestart". | The value of the PEST RSTFLE variable was incorrectly supplied. | Edit the PEST control file, altering the value for the RSTFLE variable. |
| Unknown parameter group; line *n* of file *filename*. | A parameter in the "parameter data" section has been assigned to a group which was not defined in the "parameter groups" section. | Assign the parameter to a previously defined parameter group. |
| Untransformed parameter referenced as log-transformed in prior information. Parameter *parname* on line *n* of file *filename*. | An untransformed parameter must be referenced as such in each item of prior information which cites that parameter. | Alter the parameter's PARTRANS value to "log" or reference that parameter as untransformed in the prior information equation. |
| Unrecognized decimal point indicator: line *n* of file *filename*. | An invalid character string was supplied for the variable DPOINT in the PEST control file. | Edit the PEST control file, supplying a valid value for the variable DPOINT. |
| Unrecognized derivative information; line *n* of file *filename*. | One of the variables supplied in the "parameter groups" section of the PEST control file has an unrecognizable value. | Edit the "parameter groups" section of the PEST control file and rectify the error. |
| Unrecognized parameter change limit. Line *n* of file *filename*. | The PARCHGLIM variable supplied for at least one parameter is incorrect. | Edit the PEST control file and rectify the parameter change limit. |
| Unrecognized observation group: line *n* of file *filename*. | An observation group has been cited in the "observation data" section of a PEST control file that has not been cited in the "observation groups" section of this file. | Edit the PEST control file, adding the observation group to the "observation groups" section of the PEST control file. |
| Unrecognized parameter name "*parname*": line *n* of file *filename*. | A parent or tied parameter name listed in the linkages subsection of the "parameter data" section has not been previously defined as a parameter. | Define the parameter or alter the incorrect name to the name of a defined parameter. |
| Unrecognized parameter precision type: line *n* of file *filename*. | An invalid character string was supplied for the variable PRECIS in the PEST control file. | Edit the PEST control file, supplying a valid value for the variable PRECIS. |

| | | |
|---|---|---|
| Unrecognized parameter transformation/tied/fixed information. Line *n* of file *filename*. | The PARTRANS variable for at least one parameter supplied in the PEST control file is incorrect. | Edit the PEST control file and rectify the transformation status of the parameter. |

**Table 20. Errors pertaining to the PEST control file.**

| **Error message** | **Reason for error** | **User's response** |
|---|---|---|
| Backwards move to tab position: line *n* of model output file *filename*. | Following its previous instruction, PEST has advanced further along a line of model output than the location of the tab position indicated in the present instruction. | Establish why the model output file is different from expected, or alter the instruction file until the model output file is read correctly. |
| Blank parameter space: line *n* of template file *filename*. | Only whitespace exists between two parameter delimiters at one location in a template file. | Edit the template file, placing the name of a parameter between the parameter delimiters. |
| Cannot find observation *obsname*: line *n* of model output file *filename*. | Either the model output file is different from expectations or the instruction set is incorrect. | Ascertain the reason why the model wrote an unanticipated output file, or make corrections to the instruction set. |
| Cannot interpret instruction for reading model output file. Instruction line follows:- | An instruction is incorrectly provided in an instruction file. | Alter the instruction file to provide the correct syntax. |
| Cannot read line advance item. Instruction line follows:- | A number signifying the number of lines that must be jumped when reading a model output file cannot be read. | Edit the instruction file, rectifying the error. |
| Cannot read tab position. Instruction line follows:- | The number signifying the character position to which the instructional "cursor" should advance cannot be read. | Edit the instruction file, rectifying the error. |
| Continuation character must not be first instruction on line. Instruction line follows:- | The user has provided an instruction line with incorrect syntax. | Edit the instruction file, rectifying the syntax error. |
| Error reading observation *obsname*: line *n* of model output file *filename*. | Either the model output file is different from expectations or the instruction set is incorrect. | Ascertain the reason why the model wrote an unanticipated output file, or make corrections to the instruction set. |
| Error writing parameter *parname* to model input file; internal error. | The subroutine which maximizes parameter significance in a confined space has encountered an error condition. | Report the error to the programmer. |

| | | |
|---|---|---|
| Error writing parameter *parname* to model input file; exponent too large for single precision protocol. | The magnitude of the current value of a parameter is greater than about $10^{38}$. | Alter the value of the PRECIS variable to "double" in the PEST control file. |
| Error writing parameter *parname* to model input file; exponent too large for double precision protocol. | The magnitude of the current value of a parameter is greater than about $10^{308}$. | Adjust the parameter's SCALE value to decrease the parameter's size. |
| Error writing parameter *parname* to model input file. Template field width too small to represent current value; number too large or too small to be represented with any precision. | The current value of a parameter is too large to fit into the space allowed for it in the template file. | Increase the parameter space width if the model input file protocol allows it. |
| File *filename* does not have the correct instruction file header. | The pertinent file does not begin with the string "pif #" where "#" is a marker delimiter. | Edit the instruction file and add the string to the top of the file. |
| File *filename* does not have correct template file header. | The pertinent file does not begin with the string "ptf #" where "#" is a parameter delimiter. | Edit the template file and add the string to the top of the file. |
| First instruction in file *filename* cannot start with a continuation character. Instruction line follows:- | The first instruction in the pertinent instruction file has incorrect syntax. | Edit the instruction file, rectifying the syntax error. |
| Missing marker delimiter. Instruction line follows:- | There is an error in the syntax of an instruction involving a marker delimiter cited in an instruction file. | Edit the instruction file, rectifying the error. |
| Parameter *parname* not cited in PEST control file: line *n* of template file *filename*. | A parameter name is cited in a template file. However the name of the same parameter is not cited in the PEST control file. | Add the name of the parameter, as well as data pertaining to that parameter, to the PEST control file. |
| Parameter *parname* not referenced in any template file. | A parameter is cited in the PEST control file which is not cited in any template file. | Edit the template file corresponding to the model input file to which the parameter must be written, citing the name of that parameter. |
| Parameter space less than three characters wide; line *n* of template file *filename*. | A parameter space width defined in the template file is less than three characters wide. | Increase the parameter space width if the model input file protocol allows. it. |
| Tab moves beyond end of line: line *n* of model output file *filename*. Instruction line follows:- | Either the model output file is different from expectations or the instruction set is incorrect. | Ascertain the reason why the model wrote an unanticipated output file, or make corrections to the instruction set. |
| Unable to find expected whitespace, or whitespace precedes end of line: line *n* of model output file *filename*. | Either the model output file is different from expectations or the instruction set is incorrect. | Ascertain the reason why the model wrote an unanticipated output file, or make corrections to the instruction set. |

| | | |
|---|---|---|
| Unable to find secondary marker: line *n* of model output file *filename*. | Either the model output file is different from expectations or the instruction set is incorrect. | Either ascertain the reason why the model wrote an unanticipated output file, or make corrections to the instruction set. |
| Unbalanced parameter delimiter at line *n* of template file *filename*. | A parameter delimiter preceding a parameter name is not balanced by a delimiter after the parameter name. | Edit the template file, adding the parameter delimiter. |
| Unexpected end to model output file: instruction line follows:- | In following a set of instructions, PEST unexpectedly encountered the end of a model output file. | Either ascertain the reason why the model wrote an unanticipated output file, or make corrections to the instruction set. |

**Table 21. Errors in template and instruction files, or in reading/writing model input/output files.**

| **Error message** | **Reason for error** | **User's response** |
|---|---|---|
| A covariance matrix must not be supplied for observation group "predict" when PEST is run in predictive analysis mode. | When PEST is run in predictive analysis mode the single member of the observation group "predict" is maximized or minimized, not compared with field data. | Edit the PEST control file, removing the name of the observation covariance file from beside the observation group "predict" in the "observation groups" section. |
| Cannot calculate eigenvectors of covariance matrix supplied for observation group *obsgroup*. | The covariance matrix supplied for observation group *obsgroup* is not a legal covariance matrix. | Edit the observation covariance matrix supplied for observation group *obsgroup*. |
| Error encountered in reading covariance matrix file *filename*. | The data provided in the observation matrix file *filename* cannot be read as a sequence of numbers. | Edit the observation covariance matrix file and rectify the error. |
| File *filename* contains an illegal covariance matrix. | The matrix contained in an observation covariance matrix file is illegal. | Edit the observation covariance matrix file and rectify the error. |
| If a covariance matrix is supplied then an observation group can belong to observations or prior information but not both. | A covariance matrix cannot be assigned to an observation group comprised of both observations and prior information equations. | Assign either the observation data or prior information to a different observation group and provide a separate covariance file for the new group. |
| Unexpected end encountered to covariance matrix file *filename*. | There are insufficient entries in the observation covariance matrix provided in file *filename*. | Edit the observation covariance matrix file and rectify the error. |

**Table 22 Errors in Observation Covariance Matrix Files.**

| Error message | Reason for error | User's response |
|---|---|---|
| Cannot allocate sufficient memory for least squares optimization. | The number of parameters and/or observations involved in the parameter estimation process is too large. | Reduce the number of parameters or observations, or run the case on a computer with more memory. |
| Cannot calculate derivative for parameter *parname*. Increment calculated as fraction of maximum parameter in group, which is zero. No lower increment limit provided. | Increment type for parameter group to which parameter belongs has been specified as "rel_to_max". However the maximum parameter value in group is zero. | Provide non-zero parameter absolute lower bound for pertinent parameter group (i.e. variable DERINCLB) in the "parameter groups" section of the PEST control file. |
| Cannot calculate derivative for parameter *parname*. Log-transformed parameter is zero or negative. | Incorrect initial value has been supplied for the parameter. | Provide a positive initial value for the parameter, or change its PARTRANS value from "log" to "none". |
| Cannot calculate derivative for parameter *parname*. Parameter zero so increment (calculated as fraction of parameter) also zero. No lower increment limited provided. | Increment type for parameter group to which parameter belongs has been specified as "relative". This results in a zero increment when the parameter is zero. | Provide non-zero parameter absolute lower bound for pertinent parameter group (i.e. variable DERINCLB) in the "parameter groups" section of the PEST control file. |
| Cannot calculate derivative for parameter *parname*. Parameter increment zero to precision allowed by parameter template field width. | The current value of a parameter is indistinguishable from its incremented counterpart when written to a space of limited width on the model input file. | Increase the parameter space width in the pertinent template file. If the model does not allow this, increase the parameter derivative increment. |
| Cannot proceed. All parameters are either fixed, frozen or held. | There are no adjustable parameters, or PEST has fixed all adjustable parameters at their upper/lower bounds. | Denote certain parameters as adjustable and/or alter parameter bounds. |
| Cannot restart; a PEST control file has been altered since saving restart data. | An attempt was made to re-start a previously interrupted PEST run. However the PEST control file has been altered. | Change the PEST control file back to the way it was before the previous run was commenced. |
| Cannot restart; PEST was not instructed to record restart file in previous run. | The RSTFLE variable in the PEST control file was set to "norestart". | Recommence the previous PEST run from the beginning of the parameter estimation process. |
| Cannot restart; previous PEST run was not halted prematurely. | The previous PEST run was terminated by PEST due to objective function or parameter convergence. | Use the PARREP utility to build a new PEST control file using parameter values optimized from the previous PEST run. Then start PEST again based on the new PEST control file. |

| | | |
|---|---|---|
| Cannot restart: previous PEST run was in predictive analysis mode. | An attempted is being made to restart a previous predictive analysis run in parameter estimation or regularization mode. | Alter the PEST control file, setting PESTMODE to the correct mode. |
| Cannot restart: previous PEST run was not in predictive analysis mode. | An attempt is being made to restart a previous parameter estimation or regularization run in predictive analysis mode. | Alter the PEST control file, setting PESTMODE to the correct mode. |
| Cannot restart: previous PEST run was in regularization mode. | An attempt is being made to restart a previous regularization run in parameter estimation or predictive analysis mode. | Alter the PEST control file, setting PESTMODE to the correct mode. |
| Cannot restart: previous PEST run was not in regularization mode. | An attempt is being made to restart a previous parameter estimation or predictive analysis run in regularization mode. | Alter the PEST control file, setting PESTMODE to the correct mode. |
| Cannot restart with "/j" option: unable to read Jacobian matrix from file *filename*. | The previously stored Jacobian matrix file has been deleted. | Re-start the PEST run with the "/r" switch rather than the "/j" switch. |
| Command line argument should provide filename base for current case. | The command to run PEST has been provided without a command-line argument. | Re-run PEST, providing the name of the PEST input file on the command line. |
| Improper command line switch. | An improper command line switch was entered from the keyboard. | Re-type the command, using the correct syntax. |
| Improper command line syntax. | The command used to start a PEST run was not syntactically correct. | Re-type the command, using the correct syntax. |
| Incorrect NPAR or NOBS value on first line of derivatives file *filename*. | The number of parameters and observations for which derivatives are supplied by the model does not accord with those in the current parameter estimation problem. | The model algorithm for computing parameter derivatives for the current parameter estimation case must corrected. |
| Internal error in calculating derivative for parameter *parname*. | In most cases this is caused by a parameter initial value which is not between its upper and lower bounds. | Edit the PEST control file, adjusting the parameter's upper and/or lower bound as appropriate. |
| Jacobian matrix stored in file *filename* is not current. | An attempt was made to re-commence a PEST run using the "/j" command-line switch. When PEST tried to read its previous Jacobian matrix, a discrepancy was encountered with the present data set. | Re-commence PEST execution without the "/j" command-line switch. |

| | | |
|---|---|---|
| Varying one or more parameters has no effect on model output. | Model outputs are completely insensitive to the value of at least one parameter. | Change initial parameter values, increase parameter derivative increments, or hold offending parameters fixed. |

**Table 23. General PEST errors.**

## 5.6 Stopping or Pausing PEST Execution

A number of tiny utility programs named PSTOP, PSTOPST, PPAUSE and PUNPAUSE will be provided with PEST. These will allow the user to "send a message to PEST", the nature of the message being listed in Table 24. These should be run from another terminal window, opened in the PEST working directory while PEST is running. They will each write a small file containing a coded instruction to PEST. PEST will look for this file after every model run.

| **Command** | **Action taken by PEST** |
|---|---|
| pstop | Ceases execution immediately without recording optimised parameter values or any statistics based on these. |
| pstopst | Ceases execution. However, before doing so, PEST will record optimised parameter values and all statistics based on these and corresponding residuals to its normal output files. |
| ppause | Pauses execution. |
| punpause | Re-commences execution after a pause. |

**Table 24. Commands used to stop and pause PEST execution.**

## 5.7 Dialogs

No dialog boxes occur within the user interface to PEST or any of its utilities.

## 5.8 Online Help

Because it was designed to perform a discrete numerical task, and to terminate execution when it finishes that task, no online help is available through the PEST user interface.

# 6. System Interfaces

All interfacing between PEST and the system under which it runs will take place through software generated by the FORTRAN compiler used to create the PEST executable program.

When PEST needs to run the model, it will do so through invocation of a "system" subroutine or function call. Though not standard FORTRAN functionality, such a subroutine or function is available through all FORTRAN compilers.

The volume and frequency of data transfer by functional transactions will depend entirely on the simulation model with which PEST is being used. However, as the nature of PEST's interface with that model is simply to write its input files and read its output files, data volumes involved in this process will be no more than those required by the model itself. Hence the use of PEST will not place any requirements on the system beyond those already placed by the simulation model with which it is working (i.e. its "loading factor" with respect to the model will be unity).

Correct data transfer between PEST and the model will be verified if the model runs correctly. If the model fails to run after its input files were written by PEST, an error message will be written to the screen and run record file by PEST, stating that the model's output file cannot be found. The user will then be able to ascertain any breeches in data transfer protocol that have arisen through inspecting the model input file generated by PEST. Similarly, if PEST is unable to read a model output file, the nature of the failure will be apparent from the PEST error message that is displayed as a result of this occurrence. The user will then be able to rectify the problem through making an appropriate change to the pertinent PEST instruction file. (Note that all data transfer formats have been specified in detail in Section 4 of this document.)

Data validation will take place using the utility programs described in Section 3.18 of this document.

As is explained in Section 3.17, parallelization of model runs will be achieved through "message files" in a system-independent manner. The model will actually be run by a (FORTRAN-coded) "slave program" through a system call at the receipt of a signal conveyed to it by PEST through a message file. Hence there is no requirement for the use of any network communication protocols such as TCP/IP, modem connection, ppp, etc.

# 7. Security

PEST will be used in the calibration of existing models against existing data sets, or those to be gathered in the normal course of their work by personnel of the Geoanalysis Group. Hence use of PEST will introduce no new security and access issues beyond those already in place. In short, security issues are only pertinent to the existing simulation models and the data upon which they are based. Because PEST does not "create" any data, nor alter the specifications of any simulation model (simply "adding to the value" of both of these), it is "transparent" from a security point of view.

No security devices (for example hardware locks) are required for the use of PEST.

# 8. Data and Logical Model

PEST is designed to undertake complex mathematical computations according to algorithms already described in detail herein. It is designed to work in conjunction with an existing model to enhance the capabilities of that model in processing real-world data.

Issues concerned with data transfer in and out of PEST have been covered in detail in Section 4 of this document. All such data transfers take place through the medium of user-prepared ASCII files, or ASCII files written by PEST for perusal by the user. Communications between PEST and the model with which it is working will also take place through ASCII files, with the protocol for these files being set by the model itself.

PEST does not exchange data with any external process other than the model, or with any external instrumentation. Furthermore, the nature of the numerical tasks carried out by PEST and the simulation model is such that intensive numerical computations, rather than the handling of large volumes of data, are the focus of its calculations.

# 9. References

AP-SI.1Q, Rev. 2, ICN 2. *Software Management.* Washington, D.C.: U.S. Department of Energy, Office of Civilian Radioactive Waste Management. ACC: MOL.19991214.0627.

Software Installation Test Plan for PEST Version 3.5, 10289-ITP-3.5-00.

Software Requirements Document for PEST Version 3.5, 10289-RD-3.5-00.

Validation Test Plan for PEST Version 3.5, 10289-VTP-3.5-00.

Bard, Jonathon, 1974. Nonlinear parameter estimation. Academic Press, NY. 341p.

Cooley, R.L. and Naff, R.L., 1990. Regression modeling of ground-water flow: U.S. Geological Survey Techniques in Water-Resources Investigations, book 3, chap B4, 232p.

Cooley, R.L. and Vecchia, A.V., 1987. Calculation of nonlinear confidence and prediction intervals for ground-water flow models. Water Resources Bulletin. Vol. 23, No. 4, pp581-599.

deGroot-Hedlin, C. and Constable, S., 1990. Occam's inversion to generate smooth, two-dimensional models from magnetotelluric data. Geophysics, Vol 55, No. 12.

Koch, K., 1988. Parameter Estimation and Hypothesis Testing in Linear Models. Springer-Verlag, Berlin. 377p.

Mikhail, E. M., 1976. Observations and Least Squares. IEP, NY. 497p.

Nash, J. C. and Walker-Smith, M., 1987. Nonlinear Parameter Estimation; an Integrated System in Basic. Marcel Dekker Inc., Monticello, NY. 493p.

Vecchia, A.V. and Cooley, R.L., 1987. Simultaneous confidence and prediction intervals for nonlinear regression models with application to a groundwater flow model. Water Resources Research, vol. 23, no. 7, pp1237-1250.

Watermark Numerical Computing, 2002. PEST: Model-Independent Parameter Estimation. User's Manual for PEST Version 5.5.

## Appendix: Conversion Plan

Users of version 3.5 of PEST should be notified of the existence of version 5.5 of PEST. All new features of PEST 5.5 will be described in the upgraded User's Manual (Watermark Numerical Computing, 2002) to be released with this version.

PEST version 5.5 will be completely backwards compatible with previous versions of PEST. Hence all PEST input files prepared for the use of version 3.5 will be read without error by version 5.5. Also, it is not necessary to rerun calibration exercises carried out with previous versions of PEST unless input data is modified or updated, or access to new PEST functionality is required.