1. Introduction

1.1 General

"HP" stands for "highly parallelized" or "high performance".

The HP suite is comprised of a growing number of programs, some of which are counterparts to programs of similar name in the PEST suite, and some of which do not have PEST suite counterparts. All of them read a PEST control file. All of them require that a model be run many times. All of them provide a non-intrusive interface with a model using template and instruction files.

This manual describes PEST_HP, as well as two programs that were written specifically to support the use of PEST_HP, namely PWHISP_HP and PCOST_HP. Other members of the HP suite are described in a separate manual.

All members of the HP suite undertake parallelized model runs using a similar protocol to that employed by BEOPEST. Their algorithms have been tuned for operation in parallel computing environments. Model runs can be undertaken on one or on many computers. However while no member of the HP suite has been programmed to undertake serial model runs, there is no reason why a single stream of model runs cannot be undertaken on a single machine. The algorithm employed by any member of the HP suite can readily adapt to whatever parallelization environment is available to it.

1.2 PEST_HP in Brief

PEST_HP is a heavily modified version of BEOPEST. Its inversion algorithm is similar to that of PEST (and hence BEOPEST). However parts of that algorithm have been altered to improve its performance when working in conjunction with big models with long run times and questionable numerical stability. Its inversion algorithm has also been tuned for increased efficiency in highly parallelized computing environments such as high performance clusters and the computing cloud.

There are some elements of PEST functionality which PEST_HP does not possess. In developing PEST_HP, the PEST source code has been renovated, and some of it removed, to make it easier to alter and debug. At the same time, those aspects of the PEST inversion algorithm that pertain to highly parameterized inversion and parallel run management have been enhanced and made considerably more efficient.

The following capabilities of PEST/BEOPEST have been eliminated in producing PEST_HP.

- PEST_HP cannot run in "predictive analysis" mode.
- It cannot read an external derivatives file.
- It cannot write a "PEST-to-model-message" file.
- It does not write a resolution data file.
- PEST_HP cannot be started using the "/p1" switch to activate conjunctive parallelisation of the first model run with those required for filling the first Jacobian matrix. However this capability has been replaced with functionality that achieves a similar outcome – see below.

- PEST_HP does not write a sequence of matrix files in which approximations to the posterior parameter covariance and related matrices are recorded during every iteration of the inversion process. (Actually this type of file is available through PEST only if the inversion process is over-determined and unregularized.)
- Manual and automatic user intervention are not supported by PEST_HP.
- PEST_HP does not record the AIC and similar information-theoretic statistics on its run record file. (These are meaningless in contexts of highly parameterized inversion.)
- On completion of an inversion process, PEST_HP does not undertake a final model run using optimised parameters.

Most of the above PEST capabilities are not widely used. Hence contexts in which PEST_HP can work are generally the same as those in which PEST (and BEOPEST) can work. PEST_HP reads a normal PEST input dataset. If a PEST control file instructs PEST_HP to undertake processing which it does not support, it ceases execution with an appropriate error message.

The following features are unique to PEST_HP. Collectively, they generally result in improved numerical performance of PEST_HP over that of BEOPEST when conducting highly parameterized inversion in a highly parallelized computing environment. They also result in improved reporting of that performance. All of these new features are discussed in this manual.

- During each iteration of the inversion process, PEST_HP calculates a series of Marquardt lambda values on which to base parameter upgrade calculations. The number of selected lambdas depends on the number of cores available for testing parameter upgrades. Each lambda defines a direction in parameter space; lambdas are selected using an enhanced version of the algorithm employed by the traditional PEST. PEST_HP then chooses a number of parameter sets along each of these directions as parameter upgrade candidates. These parameter sets are then distributed to networked cores for parallelized testing. This process keeps otherwise idle cores busy at the same time as it maximizes the chances of finding an improved parameter set, even in difficult numerical circumstances where the integrity of finite-difference-based derivatives may be compromised by model convergence difficulties.
- Optionally, a second round of parameter upgrade calculations and parallelized testing can be undertaken using a Jacobian matrix that is improved through Broyden updating.
- The efficiency of the BEOPEST parallel run manager has been improved. If requested, model runs can be abandoned and/or terminated during the parameter upgrade testing procedure if they take too long. Thus if testing of a problematic parameter set promulgates model solver convergence difficulties, the inversion process can continue without having to wait for completion of the afflicted model run.
- In order to reduce idle processor time, and hence to raise parallelization efficiency, the initial model run of the inversion process can be undertaken in a serial computing environment prior to using PEST_HP for implementation of the inversion process in a parallelized environment. This pre-inversion run also internally orders observations recorded in the PEST control file to match the reading sequence encountered in model output files. Where observation numbers are high, the efficiency of reading model output files can be greatly improved through this process.
- The efficiency of imposition of parameter bounds in highly parameterized contexts has been improved.

- PEST's sensitivity reuse functionality has been improved to better accommodate a parallel computing environment.
- PEST_HP delays the onset of higher order finite-difference derivatives calculation if it detects that imposition of parameter change limits have hindered improvements to the objective function.
- PEST_HP provides a continuous record of parallel computing efficiency; it also provides a record of parameter sets that instigate model run failure.
- In undertaking parameter estimation, PEST_HP can optionally write a "run results file" which records parameters and model outputs pertaining to all non-Jacobian model runs. In future versions of PEST_HP, the contents of this file may support construction of proxy models.
- The running of PEST_HP can be dedicated solely to production of a run results file based on parameter sets provided in a series of parameter value files. This can facilitate uncertainty analysis, sensitivity analysis and proxy model construction.
- PEST_HP can be instructed to cease execution after a user-specified time has elapsed, regardless of whether the inversion process is complete or not.
- Where multiple commands are used to run a model, or where PEST's observation re-referencing functionality is activated, user-specified numbers written to model output files can instruct PEST_HP to award zero sensitivities of respective model outputs to a parameter that is being varied incrementally for finite-difference derivatives calculation.
- Model-generated files can be distributed to all computing nodes (possibly after local processing) prior to the commencement of each iteration of the inversion process; in some circumstances this can lead to considerably enhanced model execution speeds.
- A PEST_HP control file can cite new parameter types, namely "secondary parameters". Use of secondary parameters can replace the PAR2PAR utility as a means of transforming parameters that are estimated through the inversion process to those that are actually used by the model.
- PEST_HP provides a novel means of obtaining a Jacobian matrix based on randomized finite differences. This methodology is still in its experimental stages.
- User-specified elements of a Jacobian matrix can be strategically set to zero. This can raise the integrity of a Jacobian matrix that has been filled through random parameter differencing.
- Where a Jacobian matrix is known to possess a blocky structure, PEST_HP can increment multiple parameters simultaneously. This, together with its Jacobian blanking functionality, can considerably reduce the model run burden of filling a Jacobian matrix in some inversion circumstances.
- A new "quick regularisation" option is available.

Meanwhile, the following (and many other) advanced features of PEST/BEOPEST are retained in PEST_HP.

- SVD-assisted inversion can be employed for estimation of parameters, and for production of calibration-constrained random parameter sets using the null space Monte Carlo methodology.
- One objective function can be traded off against another by running PEST_HP in "pareto" mode.

- Inverse problem solution strategies based on Gaussian elimination, singular value decomposition and LSQR are all supported.
- Multiple options are provided for calculation of regularisation weights, and for determination of comparative inter-regularisation-group weighting when applying Tikhonov regularisation.
- Different command lines can be used to run the model in calculating derivatives with respect to different parameters.
- Observation re-referencing supports use of one or a number of simplified models for calculation of derivatives, while retaining a complex model for testing parameter upgrades.
- Alterations made to members of the PEST suite (versions 15 and above) to promote greater inter-operability with members of the PEST++ suite have also been made to members of the HP suite. These include the use of comment lines in a PEST control file. See PEST documentation for details.

PEST_HP can be used interchangeably with PEST and BEOPEST. It requires no input data in addition to that required by the normal PEST; hence a PEST input dataset is also a PEST_HP input dataset. However a number of PEST_HP-specific control variables, and three sections, can optionally be added to a normal PEST control file to construct a PEST_HP control file. PEST and BEOPEST will cease execution with a pertinent error message if they encounter these new variables and sections. Conversely, PEST_HP will cease execution with an appropriate error message if a PEST input dataset requests functionality which is no longer available in PEST_HP.

Note that the binary restart files written by PEST/BEOPEST on the one hand, and by PEST_HP on the other hand, are not compatible with each other. Hence an inversion process which is commenced by one of these programs cannot be completed by the other.

1.3 Terminology

The “master/slave” terminology employed by BEOPEST has been replaced by new nomenclature. The “master” is now the “manager”. A “slave” is now an “agent”. All members of the HP suite (including PEST_HP itself) employ a common agent; this is named AGENT_HP. The running of PEST_HP and AGENT_HP are described below.

A manager has two roles. Firstly, it undertakes the set of numerical tasks, and implements the set of numerical algorithms, that define it. Secondly, it commissions model runs that are required by its algorithm. Each of these runs is undertaken under the supervision of an agent. Many agents can exist. The manager allocates each required model run to the fastest available agent, and monitors all agents for completion of model runs. It can accommodate late and overdue model runs and missing agents.

The software that actually manages model runs is not a discrete executable program. Rather it is a set of functions and subroutines that are called by all members of the HP suite. Hence while each member of the HP suite performs a different numerical task, all members of the HP-suite call the same run management functions and subroutines. Because of this, control variables featured in a PEST control file which govern model run management have the same roles for all members of the HP suite.

Each agent must operate from a separate folder. This folder must contain the PEST control file for the current problem, all template and instruction files cited in the PEST control file, as well as all files that are required to run the model. See documentation of BEOPEST for further details.

The version number of AGENT_HP must be the same as that of the HP suite manager program with which it interacts. If this is not the case, the manager will detect this inconsistency and cease execution with an appropriate error message.

1.4 Using PEST_HP

1.4.1 Starting PEST_HP

Use of PEST_HP is very similar to that of BEOPEST. To run the PEST_HP manager based on the contents of the PEST control file *case.pst*, start it using the command

```
pest_hp case /h :nnnn
```

where *nnnn* is a port number (for example 4004). Note the space character between the “/h” string and the following colon.

Unlike BEOPEST, PEST_HP does not operate as both a manager and an agent. Instead a general HP named AGENT_HP supervises each local model run. AGENT_HP can be used with all members of the HP suite. To run AGENT_HP, use the command

```
agent_hp case /h hostname:nnnn
```

where *hostname* is the name of the machine on which PEST_HP (or any other member of the HP suite) is running, and *nnnn* is the port number employed by PEST_HP (as provided on its command line). Alternatively use the IP address of the machine on which PEST_HP is running instead of *hostname*. (Only TCP/IP version 4 is supported at the time of writing.)

Execution of many agents can be initiated in the manner described above. As stated above, each parallel run agent must operate from a different folder. If desired, one of these folders can be the same as that employed by a HP suite manager (e.g. PEST_HP). Execution of an agent can be initiated either before or after that of PEST_HP.

If only a single agent is employed, and if PEST_HP and the agent are both operating from the same folder, then a user may wish to initiate execution of both PEST_HP and AGENT_HP using a single command. This command initiates execution of a batch script whose contents are shown in figure 1.1. The file containing this script must be prepared by you, the user. Suppose that you name this file *run_pest_hp.bat*. Then execution of PEST_HP and a single agent can be initiated using the command

```
run_pest_hp case
```

where *case.pst* is the name of the PEST control file which defines the current inversion problem.

```
echo agent_hp %1 /h %computername%:4004 > agent.bat
start agent.bat
pest_hp %1 /h :4004
```

Figure 1.1 A batch file that can initiate execution of both PEST_HP and a single agent.

In the above script *computername* is an environment variable that returns the name of a computer. The script depicted in figure 1.1 can be easily modified for initiation of multiple agents running in multiple folders.

1.4.2 Termination of PEST_HP

Operation of PEST_HP can be terminated in the same manner as that of PEST and BEOPEST. This can be done brutally by pressing the <Ctl-C> keys when focussed on the PEST_HP window. Alternatively, the PSTOP or PSTOPST command can be issued from another window open to PEST_HP's working folder.

A halted PEST_HP run can be resumed using the "/s" switch. This switch must be placed before the "/h" switch in the PEST_HP command line. Hence to recommence an interrupted PEST_HP run based on file *case.pst*, use the command

```
pest_hp case /s /h :nnnn
```

The "/s" switch is not required when recommencing execution of AGENT_HP agents.

1.4.3 Other PEST_HP Command Line Switches

The "/i" command line switch and the "/t" command line switch have the same roles for PEST_HP as they do for BEOPEST. The "/i" switch informs PEST_HP that it must read a Jacobian matrix file (i.e. a JCO file) to obtain the Jacobian matrix for use in the first iteration of the inversion process; if started using this switch, PEST_HP prompts for the name of the JCO file which it must read. The "/t" switch allows a user to associate a text string with a particular PEST_HP run; see PEST documentation for details.

A new switch, namely "/hpstart", has been introduced for the use of PEST_HP, as well as for that of PEST; it enables the latter to expedite the efficiency of an ensuing PEST_HP run. When employed on the PEST_HP command line, the "/hpstart" option instructs PEST_HP to read a "hp starter file" named *case.hp* (where the PEST control file is named *case.pst*). This file contains information previously recorded during a dedicated PEST run, also initiated using the "/hpstart" command line option, in which the model is run only once. The information contained in file *case.hp* saves PEST_HP from having to conduct an initial model run to calculate the initial objective function and initial model-calculated reference values associated with observations comprising the calibration dataset. Hence, immediately upon commencement of its execution, PEST_HP can initiate parallelization of model runs in order to fill the initial Jacobian matrix. At the same time, where observations number in the tens or hundreds of thousands, PEST_HP's reading of the model-generated counterparts of observations from model output files is accelerated, as observations will have been ordered internally to PEST_HP to match their occurrence in these files. This is further discussed later in this manual.

1.5 An Alternative Version of PEST_HP

Two versions of the PEST_HP executable program are provided. One is named *pest_hp.exe*. The other is named *pest_hp_mkl.exe*. The latter executable program is linked to the Intel Maths Kernel Library. At the same time, the programming associated with a number of numerically-intensive parts of the PEST_HP algorithm have been altered to take advantage of the efficiencies offered by this library. Where the number of parameters and/or observations is large, this can result in much faster execution of PEST_HP.

If you use *pest_hp_mkl.exe* then a DLL (i.e. a dynamic linked library) named *libiomp5md.dll* must be situated in the folder from which you run *pest_hp_mkl.exe*. Alternatively, it must reside in a folder that is accessible through your computer's PATH environment variable. Many users will already have this DLL resident on their machines. However, because some will not, it is supplied with PEST_HP.

1.6 Parallel Run Management File

As is discussed in PEST documentation, Parallel PEST (a version of PEST which uses so-called "message files" for communication between managers and agents) requires that the user construct a parallel run management file prior to initiating its execution. This file must possess the same filename base as that of the PEST control file, but its extension must be "*.rmf*". BEOPEST does not require a file of this type. However, if this file is present in the directory from which it is run, BEOPEST will read values for the optional PARLAM and RUN_SLOW_FAC control variables from this file. In contrast, PEST_HP does not read a parallel run management file at all. As will be discussed below, the PARLAM variable is redundant to parallel run management undertaken by PEST_HP; if desired, a value for the optional RUN_SLOW_FAC variable can be supplied through the PEST control file.

1.7 Parallel Run Queue Files

While PEST_HP is running, and after it has completed execution, you may see files named *pest###.dap* and *pest###.dao* in its working folder. These files may be very large. The PEST_HP run manager uses these files for parallel run management. *pest###.dap* contains parameter sets that are employed in a batch of model runs; a parameter set is transferred by the run manager to one of its agents whenever a model run is commissioned. *pest###.dao* contains model-generated observations corresponding to these parameter sets. These are sent by run agents back to the parallel run manager as model runs are completed.

The PEST_HP run manager does not delete these files on termination of PEST_HP execution in case their contents are required for a restart.

1.8 When Not to Use PEST_HP

As was discussed above, PEST_HP cannot undertake serial model runs; runs are always undertaken using a run agent. There is no lower limit on the number of run agents that PEST_HP can use; it can use as few as 1. However its inversion algorithm works best where there are many parallel run agents at its disposal. Ideally there should be at least ten (but hopefully many more), as its performance can be degraded when it works with fewer agents than this.

As presently programmed, the upper limit on the number of PEST_HP agents is set to 2048. If required for a specific application, this can be altered with ease; contact the programmer.

1.9 The "PEST Whisperer"

A program named PWHISP_HP is supplied with PEST_HP. PWHISP_HP reads PEST_HP input and output files, reflects upon what it sees in those files, and records all of its thoughts in a run record file of its own which can be inspected using a standard text editor. PWHISP_HP looks for any elements in PEST_HP setup that may compromise its efficiency, or impair its ability to perform highly parameterized inversion. It suggests to the user ways in which he/she can alter a PEST_HP input dataset, or modify PEST_HP deployment, in order to obtain better results. It tries to answer questions which a modeller may ask during or after a PEST_HP run, especially if he/she is wondering whether PEST_HP could have performed better than it did.

While much of the advice provided by PWHISP_HP is just as applicable to PEST as it is to PEST_HP, PWHISP_HP can only be used with PEST_HP. It cannot be used with PEST or BEOPEST or with any other member of the HP suite. Because of novel ways in which PEST_HP uses the Marquardt lambda and performs Broyden Jacobian updating, certain aspects of the run record file written by PEST_HP are different from that written by PEST. PWHISP_HP reads the PEST_HP run record file in detail; this is how it makes its assessment of PEST_HP performance. It has not been programmed to read a run record file written by any program other than PEST_HP.

PWHISP_HP is run using the command

```
pwhisp_hp case outfile
```

where *case.pst* is a PEST control file and *outfile* is the name for the text file which PWHISP_HP must write. This command can be issued even while PEST_HP is running; you do not have to wait until PEST_HP execution is complete in order to receive advice from the PEST whisperer. If they are available, PWHISP_HP reads some or all of the following PEST_HP input/output files, recording any suggestions that it can make based on the contents of these files on its own run record file.

- the PEST control file (*case.pst*);
- the PEST run record file (*case.rec*);
- the parameter value file (*case.par*, or *base_case.bpa* if PEST_HP is conducting SVD-assisted inversion);
- the composite sensitivity file (*case.sen*);
- the intermediate residuals file (*case.rei*); and
- the Jacobian matrix file (*case.jco*).

Note the following.

- PWHISP_HP does not offer advice on PEST_HP execution if PEST is run in “pareto” mode. This is because PEST_HP’s tasks when running in this mode are very different from when running in “estimation” or “regularisation” modes.
- It is not the intention of PWHISP_HP to repeat suggestions and warnings provided by PESTCHEK. PESTCHEK should be used to check a PEST_HP input dataset before PEST_HP is run. PWHISP_HP seeks verification from the user that this has indeed been done.

1.10 The PEST_HP Cost Estimator

Prior to running PEST_HP on the cloud, costs can be estimated using a utility program named PCOST_HP. PCOST_HP reads a PEST control file to obtain details of a PEST run; it obtains the other information that it needs for cost estimation from the command line. It is run using the command

```
pcost_hp case runtime agents rate [iteration_override]
```

where

case	is the filename base of a PEST control file;
runtime	(a real number) is the estimated model run time in minutes;

agents	(an integer) is the number of agents that PEST_HP will employ;
rate	(a real number) is the cost per minute per agent for cloud rental; and
iteration_override	(an optional integer) allows you to over-ride PCOST_HP's estimate of the number of iterations that PEST_HP will undertake.

Note that the cost per agent will be less than the cost per rented machine if a number of agents use the same machine. In fact the cost per agent will be the cost per machine divided by the number of agents that use each machine.

PCOST_HP lists the outcomes of its calculations to the screen. Figure 1.2 shows an example.

```
 Estimated cost without hp starter file = $29.00
 Estimated cost with hp starter file    = $27.20

 Some specifications:-
   PEST mode of operation                                  : regularization
   Number of adjustable parameters                         : 10
   Jacobian runs per itn before switch to higher order derivs : 10
   Jacobian runs per itn after  switch to higher order derivs : 20
   Parameter upgrade testing runs per iteration            : 5
   Number of upgrade testing cycles per iteration          : 1
   Number of iterations used for cost estimation           : 10
   Number of iterations specified in PEST control file     : 30

 Possible idle agents per iteration (in units of model runs):-
   Initial model run (if not using hp starter file)        : 4
   Initial model run (if using hp starter file)            : 0
   Fill Jacobian matrix before switch to higher order derivs  : 0
   Fill Jacobian matrix after  switch to higher order derivs  : 0
   Parameter upgrade testing                               : 0
```
Figure 1.2 An example of PCOST_HP screen output.

Cost estimates provided by PCOST_HP are only approximate. It has no idea how many iterations will be required for completion of the inversion process. So it guesses that the inversion process will reach completion after 8 iterations, unless the NOPTMAX variable in the PEST control file is set to less than this, or unless the setting of the SOFTSTOPHOURS or HARDSTOPHOURS variable (see later in this document) imposes an upper time limit on a PEST_HP run. PCOST_HP's calculations are also based on the assumption that the model run time will not vary from the initial estimate provided on its command line. In practice this may not be the case, especially for highly nonlinear models which employ adaptive time stepping for which the run time may be sensitive to the values of model parameters.

Costs may be considerably greater than estimated by PCOST_HP if PEST_HP is asked to run in "regularisation" mode, the number of parameters is high, and many of these parameters hit their bounds. As is explained in documentation of PEST, this situation requires many re-formulations of the inverse problem as pertinent parameters are sequentially frozen at their bounds. Implementation of the numerical steps required to accommodate bounds enforcement may keep the PEST_HP manager busy while agents are standing idle (sometimes for up to an hour if parameter numbers are greater than 10000, observation numbers are high, and many parameters encounter their bounds). PEST_HP cannot predict this. However it does issue a warning message if adjustable parameters number over 2500 and LSQR is not being employed as a solution mechanism for the inverse problem. (Note that, as is explained above, the time taken by PEST_HP to perform numerically intensive tasks can be considerably shortened if the *pest_hp_mkl.exe* executable program is used instead of the *pest_hp.exe* executable program. The time required to compute a regularisation weight factor can also be considerably reduced if the REG2MEASRAT variable is used to control regularisation; see section 7.8 of this document.)

A warning message is also issued if the setting of the HARDSTOPHOURS or SOFTSTOPHOURS variable affects calculation of the estimated PEST_HP running cost. Warning messages are placed under the cost estimate. See figure 1.3.

```
Estimated cost without hp starter file = $120.60
Estimated cost with hp starter file    = $119.60

Warning: estimated costs are limited by HARDSTOPHOURS setting in PST file.

Warning: the manager's run time (and hence the total cost) is unpredictable
as a large number of parameters is being estimated, but LSQR is not being
used as the solution mechanism.

Some specifications:-
  PEST mode of operation                                   : regularization
  Number of adjustable parameters                          : 1100
  Jacobian runs per itn before switch to higher order derivs : 1200
  Jacobian runs per itn after  switch to higher order derivs : 2400
  Parameter upgrade testing runs per iteration             : 100
  Number of upgrade testing cycles per iteration           : 1
  Number of iterations used for cost estimation            : 8
  Number of iterations specified in PEST control file      : 30

Possible idle agents per iteration (in units of model runs):-
  Initial model run (if not using hp starter file)         : 99
  Initial model run (if using hp starter file)             : 0
  Fill Jacobian matrix before switch to higher order derivs  : 0
  Fill Jacobian matrix after  switch to higher order derivs  : 0
  Parameter upgrade testing                                : 0
```

Figure 1.3 A cost estimate affected by the HARDSTOPHOURS setting, with an accompanying warning message pertaining to use of a large number of parameters.

As was stated above, PCOST_HP does not necessarily assume that the number of iterations required for completion of an inversion process is equal to the value of the NOPTMAX variable supplied in the "control data" section of the PEST control file. NOPTMAX is often set to a high value in order to allow other control variables to establish inversion completion. PCOST_HP assumes that after 8 iterations have elapsed, the objective function will have been reduced to a value that satisfies the user (in which case he/she will stop execution of PEST_HP him/herself), or to a value that PEST_HP cannot improve (in which case PEST_HP ceases execution itself). If neither of these occurs, then the NOPTMAX setting prevails, of course. The user can provide a value other than 8 for the purpose of cost estimation by entering this number as the value of the optional *iteration_override* command line variable. However PCOST_HP will reduce *iteration_override* to NOPTMAX if it is greater than NOPTMAX.

If PEST_HP is run in "pareto" mode, PCOST_HP does not assume that 8 iterations will be required for completion of the inversion process because, under these circumstances, PEST_HP traverses the Pareto front rather than undertaking a single inversion exercise. Instead, PCOST_HP assumes that PEST_HP will complete its traversal of the Pareto front, and that the number of iterations will therefore be equal to:

$$(NUM\_WTFAC\_INC-2)*NUM\_ITER\_GEN+NUM\_ITER\_START+NUM\_ITER\_FIN$$

(See the PEST manual for definitions of these Pareto control variables.) You can override this number through selection of a value for *iteration_override* which is less than this.

Note the following.

- If PCOST_HP encounters an error in the PEST control file which it reads, it ceases execution after writing an appropriate error message to the screen.
- An entry of "na" for any specification quoted by PCOST_HP means "not applicable". The following conditions (and others) may result in a "na" condition being recorded in PCOST_HP's screen output:
    - no parameter upgrade calculations are performed because NOPTMAX in the PEST control file is set to -2;
    - the switch to higher-order derivatives is not made because there is no parameter group for which the setting of FORCEN is either "switch" or "switch_5".
- As is explained later in this document, the number of cycles of parameter upgrade testing per PEST_HP iteration can be reduced from two to one by disabling Broyden Jacobian improvement; this is achieved by setting the JACUPDATE variable to zero in the "control data" section of the PEST_HP control file.
- Use of a hp starter file is explained later in this document.