2. Marquardt Lambda Selection

2.1 The Selection Algorithm

In BEOPEST, implementation of the "partial parallelization" procedure whereby different values are selected for the Marquardt lambda, and parameter upgrades are calculated and tested using these different Marquardt lambdas, is dependent on the setting of the PARLAM parallelization control variable. According to this setting, the Marquardt lambda selection and parameter upgrade testing procedure can be serial, partially parallel, or fully parallel. Unless PARLAM is set to -9999, an initially parallel procedure can become a serial procedure if one or more parameters encounter their bounds.

The Marquardt lambda selection and parameter upgrade testing procedure undertaken by PEST_HP is always parallel; it never becomes serial, even if some parameters hit their bounds. On all iterations of the inversion process, PEST_HP selects a set of Marquardt lambdas, calculates parameter upgrades using these lambdas, and commissions a set of parallel model runs based on these upgraded parameter sets. If it has enough run agents at its disposal, it also calculates parameter upgrades corresponding to fractional lengths along the upgrade directions defined by different Marquardt lambdas. These fractions can be both greater and less than 1.0, with a fraction of 1.0 corresponding to the optimum length of the parameter upgrade vector according to an assumption of local model linearity. The line search fractions employed by PEST_HP are pre-set; however, on any particular iteration of the inversion process, PEST_HP can adjust them to reflect the distance in parameter space between current parameter values and the closest parameter bound in the direction of each upgrade vector.

Upgraded parameter sets calculated in the manner discussed above are bundled into a set of model runs that are undertaken in parallel. The multiplicity of upgraded parameter sets that require testing occupies computing cores that would otherwise be idle. PEST_HP ensures that the number of parameter upgrades that requires testing is such that the set of parallelized models runs which implement this testing can be undertaken simultaneously in roughly the same time (based on PEST_HP's assessment of the relative computing speeds of agents which are at its disposal). Once these model runs have been completed, PEST_HP processes the results, and selects the best parameter set for potential use in the next iteration of the inversion process. (Note that "best" depends on whether PEST_HP is running in "estimation", "regularisation" or "pareto" mode.)

Optionally, PEST_HP then repeats this process. However before doing so, it improves the Jacobian matrix using a Broyden update procedure informed by the results of the just-commissioned set of parallel model runs. The number of Broyden Jacobian updates which PEST_HP undertakes is set by the JACUPDATE variable supplied in the PEST control file; however it will never exceed the number of parameter upgrade directions determined by the number of Marquardt lambda values just employed (each Marquardt lambda defines a different update direction). Once this new set of parallelized model runs is complete, PEST_HP moves on to the next iteration.

During any iteration of the inversion process, PEST_HP calculates Marquardt lambda values itself. Its choice of the number of lambda values to employ depends on the number of available parallel run agents. However it will never employ less than three Marquardt lambda values, even if this is less

than the number of parallel agents available to it. Furthermore, a user can constrain PEST_HP's choice of the number of Marquardt lambdas to employ in calculating upgrades through use of the PEST_HP-specific UPTESTMIN and UPTESTLIM control variables; see later in this manual. In calculating lambda values, PEST_HP ignores variables which control the Marquardt lambda selection procedure provided on the sixth line of the PEST control file. In particular, it ignores the RLAMDA1, RLAMFAC, PHIRATSUF, PHIREDLAM and NUMLAM variables. To make its indifference to the values supplied for these variables explicit, it does not even record them on its run record file. (It is nevertheless wise to choose sensible values for these variables so that PESTCHEK does not complain if asked to review the PEST control file.)

The optional JACUPDATE variable also appears on the sixth line of the PEST control file. If this is set to zero, or omitted from the PEST control file altogether, PEST_HP will not undertake Broyden enhancement of the Jacobian matrix based on the parallelized set of model runs which were carried out to test parameter upgrades. Instead, PEST_HP immediately commences the next iteration of the inversion process wherein it initiates finite-difference calculation of a new Jacobian matrix. Alternatively, if JACUPDATE is set to *N*, then PEST_HP undertakes *N* updates of the Jacobian matrix, based on directions in parameter space corresponding to the *N* most productive Marquardt lambdas. (Updating of the Jacobian matrix in this fashion does not require any model runs. It is an internal numerical procedure that is accomplished quickly; see PEST documentation for details.) If necessary, PEST_HP reduces *N* to the number of Marquardt lambdas that was actually used during the immediately preceding round of parallelized parameter upgrade tests. If you do not know what value to supply for JACUPDATE, set it to a high number such as 999; PEST_HP will then undertake as many Jacobian updates as it can. Alternatively, omit it from the PEST control file in order to forego Jacobian updating altogether. Limited experience to date suggests that on some occasions (particularly when running PEST_HP in "pareto" mode), the second round of parallelized parameter upgrade testing based on a Broyden-improved Jacobian matrix can be worth the trouble. On other occasions it is not, particularly if the number of available parallel run agents is large; in this case there is a greater computational advantage to be gained in re-computing the Jacobian matrix altogether using the just-upgraded parameter set than in trying to improve it in the hope of calculating a better upgrade.

PEST_HP will never undertake more than two rounds of parallelized parameter upgrade testing (and hence one round of Broyden Jacobian updating). Unless there are fewer than three agents at its disposal, each set of parallelized model runs required for parameter upgrade testing will be comprised of model runs which are fewer or equal in number to the number of agents which are at its disposal. The procedure is thus designed to maximize use of otherwise idle computing cores while progressing the inversion process as quickly as possible.

2.2 Upper Upgrade Test Limit

As stated above, when calculating parameter upgrades, PEST_HP selects Marquardt lambdas itself. It also calculates intervals along the parameter upgrade vectors that are computed using these Marquardt lambdas for which new parameters will be generated and then tested using parallelized model runs. The total number of upgraded parameter sets for which model runs can be commissioned can be as high as about 130, the exact number depending on the case. However, in practice, the number of parameter upgrades that are tested is restricted to the number of active agents. If more than 130 agents are available, then those agents which are not engaged in

undertaking model runs for the purpose of testing parameter upgrades remain idle during the parameter upgrade testing procedure.

Its capacity to simultaneously test many different parameter upgrades is one of the strengths of the PEST_HP inversion algorithm. However, for some models, the number of runs devoted to upgrade testing may need to be restricted to a smaller number than PEST_HP would normally choose as it attempts to keep all agents busy. On some occasions, for some highly nonlinear models, parameter sets that are calculated using very high or very low Marquardt lambdas can induce convergence difficulties in the model's solver. Model instances which employ these parameter sets may then take a long time to run, this holding up the whole inversion process as other agents remain idle while waiting for these runs to finish. While this problem can be addressed to some extent using the RUN_ABANDON_FAC and WIN_MRUN_HOURS variables (see later in this manual), its chances of occurrence can be reduced by limiting the number of agents that are actually used in a single round of parallel upgrade testing. This limit applies to both the first round of parameter upgrade testing undertaken immediately after filling the Jacobian matrix, and to the second round of parameter upgrade testing that is commissioned after Broyden Jacobian updating has been completed. Upgrade agent limiting can be accomplished using the UPTESTLIM control variable.

The UPTESTLIM variable is optional. If present, it is placed in the "control data" section of the PEST control file – on the sixth line of this file. It can be placed anywhere after the values supplied for RLAMBDA1, RLAMFAC, PHIRATSUF, PHIREDLAM, NUMLAM, and (optionally) JACUPDATE, which also appear on this line. Suppose that you wish to limit PEST_HP to the use of no more than 50 agents when testing parameter upgrades. Then the string "UPTESTLIM=50" should be written on this line. See appendix 1 of this document for an example.

If the number of model runs devoted to parameter upgrade testing is limited in this fashion, then PEST_HP calculates Marquardt lambdas and line search factors in a way that maximizes upgrade testing efficiency while respecting this model run limit. However caution should be exercised in setting an upgrade testing limit in this way. If it is set too low, then some of the benefits of using PEST_HP will be lost.

2.3 Lower Upgrade Test Limit

Sometimes it is useful to set a lower limit on the number of parameter upgrade tests that PEST_HP undertakes, even if it does not have enough agents at its disposal to undertake the necessary model runs in parallel all at once. A user may insist, for example, that at least 20 parameter upgrade tests be undertaken even though PEST_HP may have access to 10 CPU's. He/she may do this in order to gain the benefits of the line search that PEST_HP undertakes along different parameter upgrade directions that are calculated using different Marquardt lambdas. This may prove a fruitful strategy in highly nonlinear inversion contexts.

The UPTESTMIN control variable can be used to set this minimum number of upgrade tests. Its use is almost identical to that of UPTESTLIM. Insert the string "UPTESTMIN=*N*" (where *N* must be replaced by the desired minimum number of upgrade tests) on the sixth line of the PEST control file anywhere after the values supplied for RLAMBDA1, RLAMFAC, PHIRATSUF, PHIREDLAM, NUMLAM, and (optionally) JACUPDATE.

Note that parallelization efficiency is maximized if UPTESTMIN is an integral multiplier of the number of agents that PEST_HP is using to supervise model runs.