

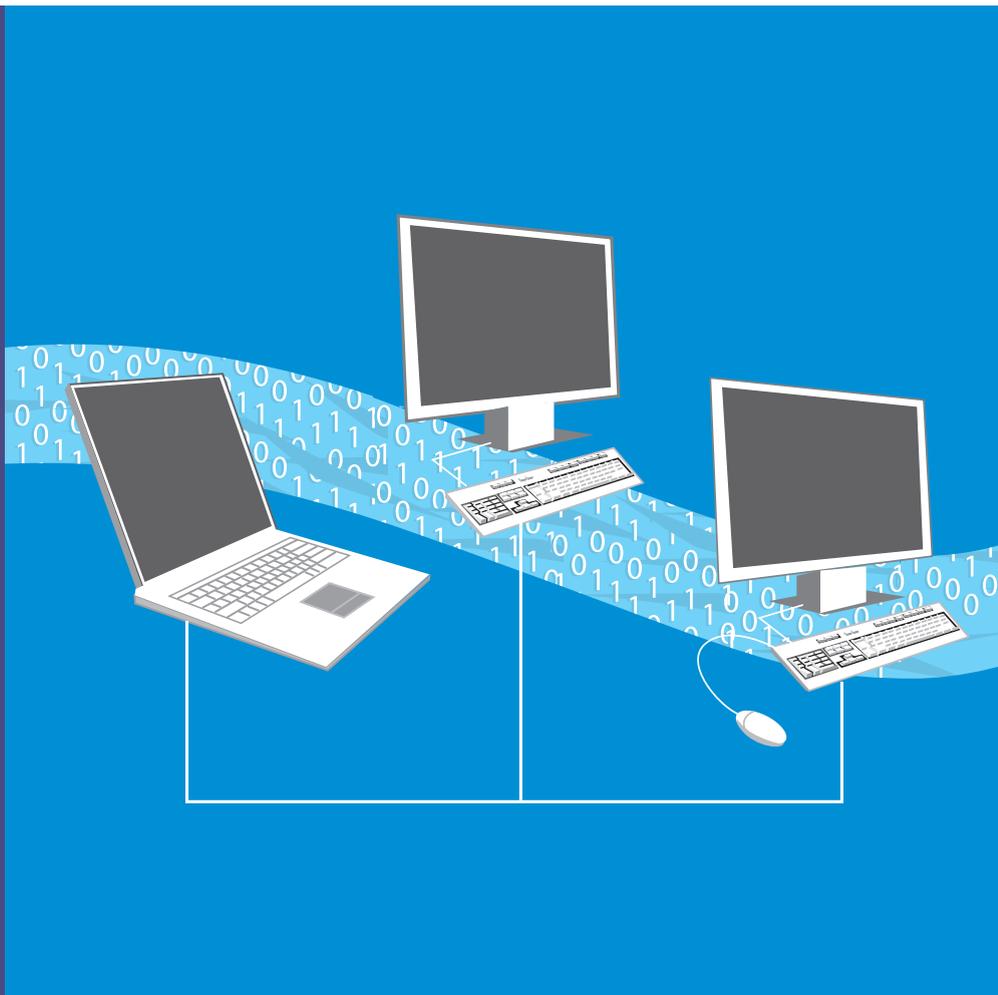


Utilization of the new features in PEST

Gusong Ruan

1
2008

R
E
P
O
R
T



Utilization of the new features in PEST

To calibrate distributed version of HBV model by
Parallel PEST

Report no. 1.08

Utilization of the new features in PEST

Published by: Norwegian Water Resources and Energy Directorate

Author Gusong Ruan

Printed by: Norwegian Water Resources and Energy Directorate

ISBN 978-82-410-0656-2

ISSN 1502-3540

Key words: Model calibration

Norwegian Water Resources and Energy Directorate
Middelthunsgate 29
PO Box 5091 Majorstua
0301 OSLO
Norway

Telefon: 22 95 95 95
Telefaks: 22 95 90 00
Internett: www.nve.no

Januarv 2008

Table of Contents

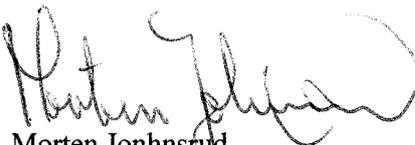
Utilization of the new features in PEST.....	1
To calibrate distributed version of HBV model by Parallel PEST	1
Table of Contents.....	3
1 Introduction.....	6
2 To calibrate distributed HBV model by Parallel PEST.....	6
2.1 What is PEST and what is Parallel PEST?	6
2.2 Why Parallel PEST?.....	7
2.3 How Parallel PEST works?	7
2.4 Preparing for a Parallel PEST run.....	8
2.4.1 Overview of the files required for a Parallel PEST run	8
2.4.2 Preparing run management file.....	9
2.5 Files in PSLAVE working directory.....	11
2.6 Starting Parallel PEST	11
2.6.1 Starting the slave.....	12
2.6.2 Starting PEST.....	12
3 Stopping and restarting PEST	12
3.1 Interrupting PEST execution	12
3.2 Restarting PEST with the “/r” switch.....	13
3.3 Restarting Parallel PEST with “/s” switch.....	13
3.4 Restarting PEST with “/j” switch.....	13
4 Testing different values of FORCEN and DERMTHD.....	14
4.1 Catchments and calibration period.....	14
4.2 Values of FORCEN and DERMTHD	14
4.3 R2 values	14
5 References	16
Appendix	17

Preface

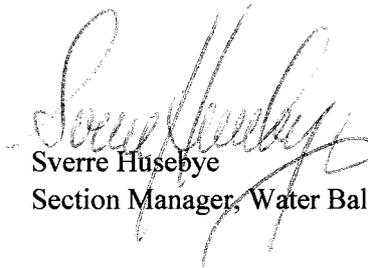
There are a range of models for available for simulation of stream discharge based on precipitation and temperature inputs. At one extreme are the purely empirical, lumped black-box models. These models identify a relationship between input and streamflow output without attempting to describe any of the mechanisms whereby this process takes place. At the other extreme are the distributed physically based models which attempt to describe the spatial distribution of storage and flow of water using equations based on principles from fluid mechanics and thermodynamics. The resulting system of partial differential equations has to be solved numerically at all points on a three-dimensional grid representation of the catchment. Between these extremes we find the majority of the hydrological simulation models in use, the conceptual models, which are often semi-distributed. They represent the various water stores (e.g. snow, interception, soil moisture, groundwater zone) and flux terms (e.g. infiltration, evapotranspiration, snowmelt, groundwater flow) with varying levels of complexity. Regardless of the complexity of the hydrological model, the mathematical and logical expressions used to describe the hydrological system contain variables and parameters. Model parameters remain constant over time or vary in a manner which may be described using physical principles or empirical relationships. Parameters either represent physically measurable properties of a catchment, or are used to describe hydrological processes. A variable may represent: (i) the state of the different stores in the hydrological system as approximated by the hydrological model; (ii) the input signal that drives the model; or (iii) the output from the model. Variables vary with time.

Hydrological models that are to be used for operational purposes must be calibrated, i.e. the values of their parameter must be determined using observed catchment input and response. Manual calibration of hydrological models is possible, but it can be time consuming and is usually not realistic for operational systems. The calibration procedure requires an automatic parameter optimization algorithm and an objective function derived from a period where all operational modes of the hydrological model can be expected to be activated. This study investigates the calibration software PEST and some possibilities for improving its performance.

Oslo, January 2008



Morten Jonhnsrud
Director, Hydrology Department



Sverre Husebye
Section Manager, Water Balance

Summary

Some interesting features in the parameter estimation software PEST have been executed. These features include Parallel PEST, stopping and restarting PEST and using different values of the character variables FORCEN and DERMTHD in PEST control file. The distributed version of HBV model is executed with Parallel PEST as well as with normal PEST, whereas the lumped version of HBV model is only executed with the normal PEST to test various settings of the character variables. Catchments Gaulfoss and Eggafoss are selected in calibration by Parallel PEST. Gaulfoss is also used in calibration by normal PEST for comparison. The runtime of Parallel PEST and normal PEST on catchment Gaulfoss are recorded. How to prepare a Parallel PEST run and how to execute Parallel PEST are presented. The ways to interrupt and to restart PEST and Parallel PEST are shown. R2 values from the calibration by different values of the character variables in PEST control file are presented.

1 Introduction

Model independent program PEST has been used to calibrate the lumped version of HBV model in many years. As the continuous development of PEST many new features are available for the improvement of the calibration. Parallel PEST and the manner to interrupt and to restart PEST are two of them. Parallel PEST makes it possible to calibrate more complex model by taking advantage of the new technology in computer science, whereas the manner to interrupt and restart PEST provides a way for user to pause and restart execution of PEST, so that the user can inspect the run record file and do possible change on control variables during the time when calibration is ongoing.

The distributed HBV model (Beldring, 2007) is developed among others to catch up the availability in grid meteorology data and to satisfy the raised need for grid runoff data. Using the existing calibration protocol with normal PEST to calibrate the distributed HBV model will result in a very long calibration time and hence to obstacle the use of the distributed HBV model. To calibrate the distributed HBV model using Parallel PEST gives the hope to solve this problem. This report presents how to run Parallel PEST on a multi-processor computer or a series of networked computers. The execution was applied on the catchment Gaulfoss with an area of 3079 square kilometers and observation of five years. It is also presented how to interrupt and restart PEST and Parallel PEST. In addition a test of different values of the character variables FORCEN and DERMTHD, which control the method used to calculate derivatives, is made with purpose to find the best values of these variables leading to improvement of the calibration. There are nine catchments involved in the test. R2 values from calibrations as well as from validations are presented for each catchment.

2 To calibrate distributed HBV model by Parallel PEST

2.1 What is PEST and what is Parallel PEST?

PEST is software for model-independent parameter estimation and uncertainty analysis, developed by S.S. Papadopoulos Associates, Inc. Parallel PEST is one of the new features included in PEST's latter version.

Parallel PEST distributes model runs across networked computers or different processors in the same computer. When model run times are larger and adjustable parameters are many, the saving in overall PEST optimization time through the use of Parallel PEST is enormous. An efficient use of Parallel PEST assumes that model run times are larger than 30 seconds and the number of the adjustable parameters is more than 3 or 4.

2.2 Why Parallel PEST?

The purpose of using Parallel PEST is to save the calibration time by taking great advantage of the computational power either in a distributed computer system or in a multi-processor computer. It is of special importance when the model is large and complex. In most cases of optimization the bulk of PEST's running time is consumed in running the model. Any time savings made in carrying out these model runs will result in dramatic enhancements to overall PEST performance.

In order to understand how much time can be saved by Parallel PEST respective to the normal PEST in model optimization, a case study has been carried out where the distributed version of the HBV model (Beldring et al., 2007) applies in the calibration by Parallel PEST and by normal PEST. The calibration catchment is Gaulfoss and the calibration period is from 1988 to 1992. The run time of the model calibrations is listed in table 1. The numbers of the slaves showed in table 1 represents the numbers of the processors where the model runs are distributed to.

Table 1. Runtime of the optimizations on the distributed version of the HBV model

Optimization software		Runtime of the model optimization
Normal PEST		15 hours 20 minutes
Parallel PEST	2 slaves	10 hours 44 minutes
	4 slaves	8 hours 4 minutes

The case study demonstrates that using of Parallel PEST results in a considerable time savings when compared with that of the normal PEST. The parallelization in model calibration by Parallel PEST provides an ideal means to solve complex optimization problems.

It needs to be noted that the runtime of the model optimization is also dependent on the percentage of the CPU occupation by other program on the computer or processor where the model optimization is undertaken. The runtimes of the optimization listed above may be slightly shorten if no other time-consuming program is running simultaneously on the same machine or longer if the CPU occupation by other time-consuming program is high.

2.3 How Parallel PEST works?

The way in which Parallel PEST carries out model run on different machine is just a simple extension of the way in which PEST carries out model run on a single machine. Before running a model on any machine Parallel PEST writes input files containing the adjustable parameters. After the model has finished execution Parallel PEST reads one or more model output files.

While Parallel PEST is able to achieve access to model input and output files residing on another machine, it cannot actually run the model on another machine. Only another program residing on the other machine can do that. This program is named PSLAVE.

Before PEST commences execution, PSLAVE must be started on each machine where model will run. It detects the commencement of PEST through reading a signal sent by PEST. It then awaits an order by PEST to commence a model run, upon the arrival of which it sends a command to its local system to start the model run.

There are two parallelization² in Parallel PEST. One is parallelization of the Jacobian matrix calculation process and another is parallelization of the Marquardt lambda testing

process. Parallel PEST activates automatically the parallelization of the Jacobian matrix calculation process. While the parallelization of the Marquardt lambda testing process should be activated by a control variable.

2.4 Preparing for a Parallel PEST run

2.4.1 Overview of the files required for a Parallel PEST run

The preparation for a Parallel PEST run is very similar to that for a normal PEST run if the models applied to both run are the same, i.e. run environment are the same. Parallel PEST requires one extra file which is run management file-“hbv.rmf”. This file has the same filename base as PEST control file but with extension “rmf”.

Table 2 lists the files needed for the lumped version of HBV model calibrated by normal PEST and the distributed version of HBV model calibrated by normal PEST and by Parallel PEST. Gaulfoss (with code 122.9) is the calibration catchment in the case demonstrated. The PEST control file in column 3 has the same protocol as that in column 2. The template files, the instruction files and the model input files in column 3 relate to the distributed version of HBV model. The description of the model input files in column 3 can be found in “Distributed HBV model” (Beldring, 2007). The run management file i.e. -hbv.rmf will be described in section 2.4.2.

Table 2. The files prepared for a Parallel PEST run and a normal PEST run.

1	2	3	4
File type	Normal PEST (Lumped HBV model)	Normal PEST (Distributed HBV model)	Parallel PEST (Distributed HBV model)
Control file	122.9.pst	hbv.pst	As in column 3
Template files	param.tpl	catchment_correction.tpl hbv_common_parameters.tpl hbv_landsurface_parameters.tpl hbv_soil_parameters.tpl	As in column 3
Instruction files	qsim.ins res.ins	122.9.0.1001.1_ins 122.11.0.1001.1_ins	As in column 3
Model input files	climcha.dat default.dat param.dat vegtype.dat param.list kommandobuffer ptq.dta	catchment_correction.dta hbv_common_parameters.dta hbv_landsurface_parameters.dta hbv_soil_parameters.dta hbv_elements.dta hbv_landscape.dta hbv_waterland.dta met_stations.dta watershed.dta control_hbv_calibration.txt	As in column 3

		obs_streamflow.var	
Run management file			hbv.rmf

2.4.2 Preparing run management file

The purpose of the Parallel PEST run management file is to inform PEST of the working directory of each slave, and of the names and paths pertaining to each model input file in which it must write and output file which it must read.

Example 1 shows the structure of a run management file. While example 2 shows an example of run management file in the case where there are two slaves.

Example 1. Structure of the Parallel PEST run management file.

```
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)
.....
```

The meaning and the value of each variable in Example 1 are listed below.

- “prf” is a symbol to identify the file as a PEST run management file.
- NSLAVE is the number of the slaves involving in the current Parallel PEST run.
- IFLETYP must be either 0 or 1. If it is 1, then all model input files containing parameters requiring optimization and output files on the various slave machines (slave directories) must be individually named (as is demonstrated in Example 2). If it is 0, it implies that the corresponding input and output files have the same name as that provided in the PEST control file across all slaves, and each set of these model input and output files lies within the PSLAVE working directory on each slave machine. Under these conditions, all the model input and output filenames can be omitted from the run management file; Example 3 shows such an abbreviated file equivalent to that of Example 2. An abbreviated run management file contains only four parts.
- WAIT is the duration of the pause undertaken by PEST and PSLAVE at certain places in communications process. Its value is expressed in seconds and must be greater than zero. If PEST and all slaves are running on the same machine a value of 0.2 seconds is normally adequate.

- If PARLAM is set to 1, partial parallelization of the lambda search is undertaken. If it is set to 0 or is omitted, then the lambda search is conducted in serial fashion using just one processor.
- SLAVNAME is the name of the slave machine (directory); any name of up to 30 characters in length can be provided.
- SLAVDIR is the PSLAVE working directory as seen by PEST.
- RUNTIME is the expected run time of the model on the respective slave. Runtimes supplied in seconds. It is better to overestimate, rather than underestimate these run times.
- INFLE is the model input file containing parameters requiring optimization. Either full pathname or abbreviated pathnames can be supplied. The number of the files is equal to the number of the template files described in PEST control file.
- OUTFLE is the model output file read by PEST. Its name should be also given full path or relative path. The number of the output files is equal to the number of the instruction files described in PEST control file.

Example 2 shows a typical run management file. There are two slaves involved in the Parallel PEST run. The names of all model input files and all output files on both slaves are supplied individually by abbreviated pathname. The duration of each pause undertaken by PEST and PSLAVE in the communication process is 0.2 second. The partial Parallel PEST is activated. The slaves are named Apple and Banana respectively. Their working directories are the subdirectories of the PEST working directory, named slave1 and slave2 respectively. The estimated run time of the model is 180 seconds on both slaves.

Example 2. A typical Parallel PEST run management file.

```
prf
2 1 0.2 1
Apple ./slave1/
Banana ./slave2/
180 180
./slave1/catchment_correction.dta
./slave1/hbv_common_parameters.dta
./slave1/hbv_landsurface_parameters.dta
./slave1/hbv_soil_parameters.dta
./slave1/hbv_01220009.var
./slave1/hbv_01220011.var
./slave2/catchment_correction.dta
./slave2/hbv_common_parameters.dta
./slave2/hbv_landsurface_parameters.dta
./slave2/hbv_soil_parameters.dta
./slave2/hbv_01220009.var
./slave2/hbv_01220011.var
```

As we see that all model input files and output files in Example 2 lie within the slave's working directories for all slaves, so that the value of IFLETYP can be set to 0 and hence the names of all input files and output files can be omitted from the run management file.

A run management file, omitting the names of all the model input files and output files, is shown in Example 3, which is equivalent to that of Example 2.

Example 3. A Parallel PEST run management file equivalent to that of Example 2, but with IFLETYP sets to zero.

```
prf
2 0 0.2 1
Apple ./slave1/
Banana ./slave2/
180 180
```

2.5 Files in PSLAVE working directory

For convenience the model input files, program to start slave and script to start model should be resided in each of the slave working directories.

Model input files:

- catchment_correction.dta
- hbv_common_parameters.dta
- hbv_landsurface_parameters.dta
- hbv_soil_parameters.dta
- hbv_elements.dta
- hbv_landscape.dta
- hbv_waterland.dta
- met_stations.dta
- watershed.dta
- control_hbv_calibration.txt
- obs_streamflow.var

Program to start slave:

- pslave

Script to start the distributed HBV model:

- start_hbv

start_hbv is a shell script to run the distributed HBV model which has following contain:

```
/home/flom/DistHbv/Bin/hbv < control_hbv_calibration.txt
```

2.6 Starting Parallel PEST

Before starting Parallel PEST the environment variable SNOWMET should be set by following command in PEST working directory as well as in slave working directories:

```
setenv SNOWMET /mnt/readonly-snowmap2/metdata
```

2.6.1 Starting the slave

At least one of the slaves should be started before starting PEST. Open one command-line window for each slave. Go to the slave working directory and then write the following command on this window to start slave.

```
./pslave
```

When slave is started, it will print out a message that asks user to enter the command to run the model. The command entered here should be exactly the same as that written in the command section of PEST control file. The following command entered in this case:

```
./start_hbv
```

2.6.2 Starting PEST

When at least one slave has been started, Parallel PEST can be started by the following command.

```
/usr/local/pest/utpakket/ppest hbv
```

At the beginning of the execution Parallel PEST commences only one model run before distributing model runs on several computers. This is the reason that only one slave should be started before starting Parallel PEST. When Parallel PEST is started the rest of the slaves can be started if not all of them started at the beginning.

3 Stopping and restarting PEST

PEST provides a set of commands to interrupt PEST. With these commands user can cease immediately the execution, cease the execution after parameter statistics being printed out or pause the execution and restart paused execution.

When execution of PEST was interrupted it can be restarted. If the character variable RSTFLE in PEST control file has been assigned the value “restart”, PEST is instructed to periodically dump the contents of its memory to a number of binary files, so that if its execution is terminated at any stage, it can be restarted later. This takes advantage of retain the work it has been done. PEST provides three ways to restart the execution. Each of which restarts the execution from different point in the optimization.

3.1 Interrupting PEST execution

At the end of every model run PEST checks for the presence of a file named *pest.stp* in its working directory. If this file is present, PEST reads the first item in the file. The value of this item determines how PEST will be interrupted.

The file *pest.stp* can be written either by using any text editor or by commands provided by PEST. Beside the window where PEST is running, another window should be open for the edit of the file *pest.stp*. The commands used to write a value in the file *pest.stp* are:

Command	Value written in <i>pest.stp</i>	the execution PEST will undertake
PUNPAUSE	0	Resume PEST execution
PSTOP	1	PEST ceases execution immediately
PSTOPST	2	PEST ceases execution after it prints out parameter statistics
PPAUSE	3	PEST pauses execution

3.2 Restarting PEST with the “/r” switch

To restart PEST with the “/r” switch will restart PEST at the beginning of the optimization iteration even if the execution was terminated at the middle of the optimization iteration. This means can be used to restart PEST and Parallel PEST. The following command is used to restart PEST.

```
pest hbv /r
```

Where *hbv* is the filename base of the PEST control file. The corresponding command to restart Parallel PEST is:

```
ppest hbv /r
```

3.3 Restarting Parallel PEST with “/s” switch

The “/s” switch can be used to restart execution of Parallel PEST at the same model run at which its execution was terminated. This switch cannot be used to restart execution of PEST.

The command to restart Parallel PEST is:

```
ppest hbv /s
```

3.4 Restarting PEST with “/j” switch

The “/j” switch can be used to restart execution of PEST and that of Parallel PEST. By this means PEST will re-commence execution at the place at which the last Jacobian matrix had just finished being calculated. It moves straight into calculation of the parameter upgrade vector and the testing of different Marquardt lambdas.

Re-commencement of PEST execution for upgrade vector re-calculation is done by running PEST using command

```
pest hbv /j
```

or, if using Parallel PEST,

```
ppest hbv /j
```

4 Testing different values of FORCEN and DERMTHD

The purpose of testing different values of the control variables is to find the way leading to as better calibration results as possible. The variables tested in this case are character variables FORCEN and DERMTHD which determine which method (two-point, three-point or the combination of both) should be employed to calculate derivatives for group members.

4.1 Catchments and calibration period

There are nine catchments involved in the testing:

- 103.40 Horghiem
- 138.1 Øyungen
- 157.3 Vassvatn
- 16.193 Hørte
- 174.3 Øvstevatn
- 2.323 Fura
- 28.7 Haugland
- 76.5 Nigardsjøen
- 97.1 Fetvatn

The calibration period is from 1981-09-01 to 2000-08-31 for all catchments. The validation period is from 1961-09-01 to 2006-08-31 for all catchments.

4.2 Values of FORCEN and DERMTHD

The values tested for variable FORCEN are always_2 and switch. Where FORCEN is set a value of “switch”, the value of the character variable DERMTHD has been supplied with “parabolic”, “best_fit” and “outside_pts” respectively.

4.3 R2 values

Table 4 shows the R2 values from the calibration while table 5 shows the R2 values from the validation. Slightly improvement can be found in the calibration results when valuable FORCEN has value “switch” comparing with that FORCEN has value “always_2”. But this improvement doesn’t leader to the same improvement in the results from validation.

Table 4 R2 values from the calibration.

Catchment code	always_2	switch		
		parabolic	best_fit	outside_pts
103.40	0.82	0.83	0.82	0.83
138.1	0.81	0.81	0.81	0.81
157.3	0.71	0.73	0.73	0.73
16.193	0.67	0.66	0.69	0.68
174.3	0.74	0.76	0.75	0.75

2.323	0.63	0.64	0.64	0.64
28.7	0.79	0.79	0.79	0.79
76.5	0.92	0.92	0.92	0.92
97.1	0.66	0.67	0.67	0.67

Table 5 R2 values from the validation.

Catchment code	always_2	switch		
		parabolic	best_fit	outside_pts
103.40	0.80	0.81	0.80	0.80
138.1	0.69	0.69	0.69	0.69
157.3	0.70	0.72	0.71	0.72
16.193	0.66	0.65	0.67	0.67
174.3	0.68	0.67	0.67	0.68
2.323	0.61	0.61	0.61	0.60
28.7	0.72	0.72	0.72	0.72
76.5	0.91	0.91	0.91	0.91
97.1	0.46	0.46	0.46	0.46

5 References

Beldring, S., Engeland, K., Roald, L.A., Sælthun, N.R., Voksø, A. 2003. Estimation of parameters in a distributed precipitation-runoff model for Norway. *Hydrology and Earth System Sciences* 7, 304-316.

Beldring, S., 2007. Distributed HBV model. Unpublished manuscript, 23 pp.

Appendix

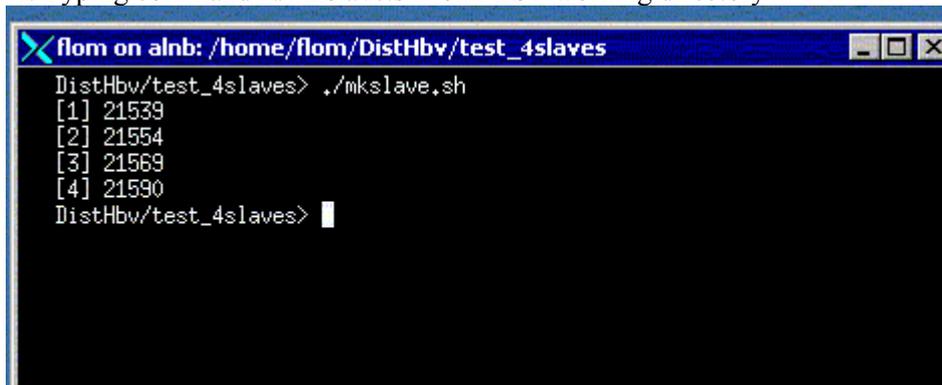
Example 1. To Run a Parallel Pest

When template and instruction files as well as a PEST control file are prepared and all model input files reside in the PEST working directory, Parallel Pest can be run by the following order:

1. Start at least one of the PSLAVE executions by typing command “pslave” on the PSLAVE working directory. “pslave” is a program in PEST program package.
2. Start PEST by typing command “ppest” on PEST working directory.
3. Start rest of the PSLAVE executions.

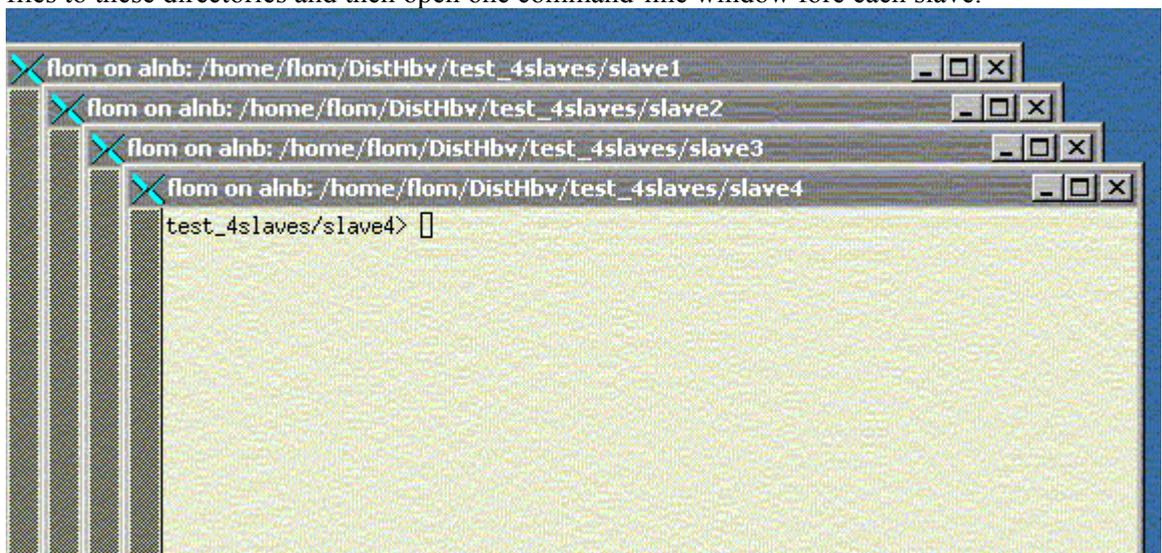
The following shows an example to run Parallel PEST. PEST working directory in the example is “DistHbv/test_4slaves”. There are four PSLAVE working directories named “slave1”, “slave2”, “slave3” and “slave4” respectively. They are subdirectories of the PEST working directory. The file named “mkslave.sh” is a script to copy all model input files to the slave working directories and open one command-line window for each slave.

1. Typing command “./mkslave.sh” on PEST working directory

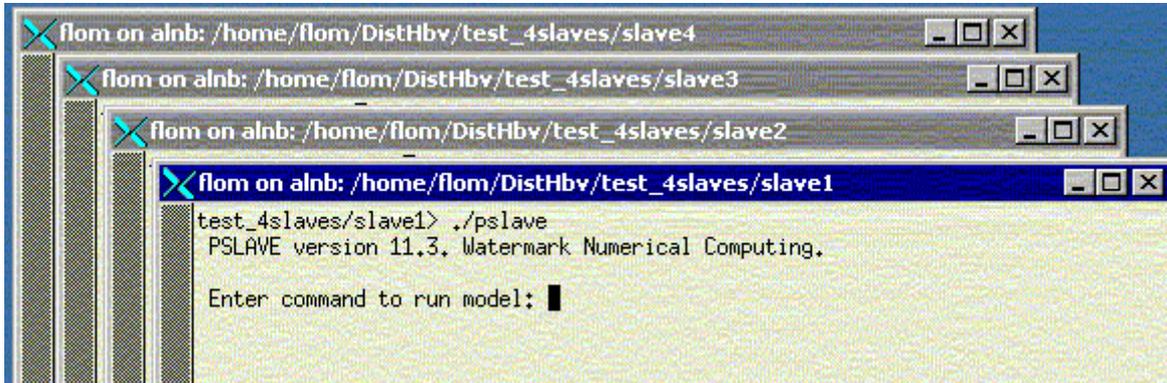


```
flo on alnb: /home/flo/DistHbv/test_4slaves
DistHbv/test_4slaves> ./mkslave.sh
[1] 21539
[2] 21554
[3] 21569
[4] 21590
DistHbv/test_4slaves> |
```

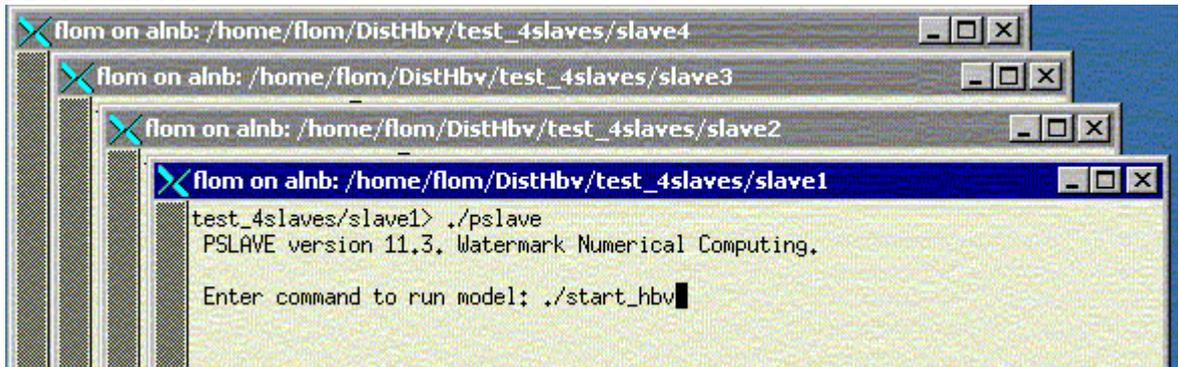
2. Script “mkslave.sh” creates four PSLAVE working directories, copies all model input files to these directories and then open one command-line window fore each slave.



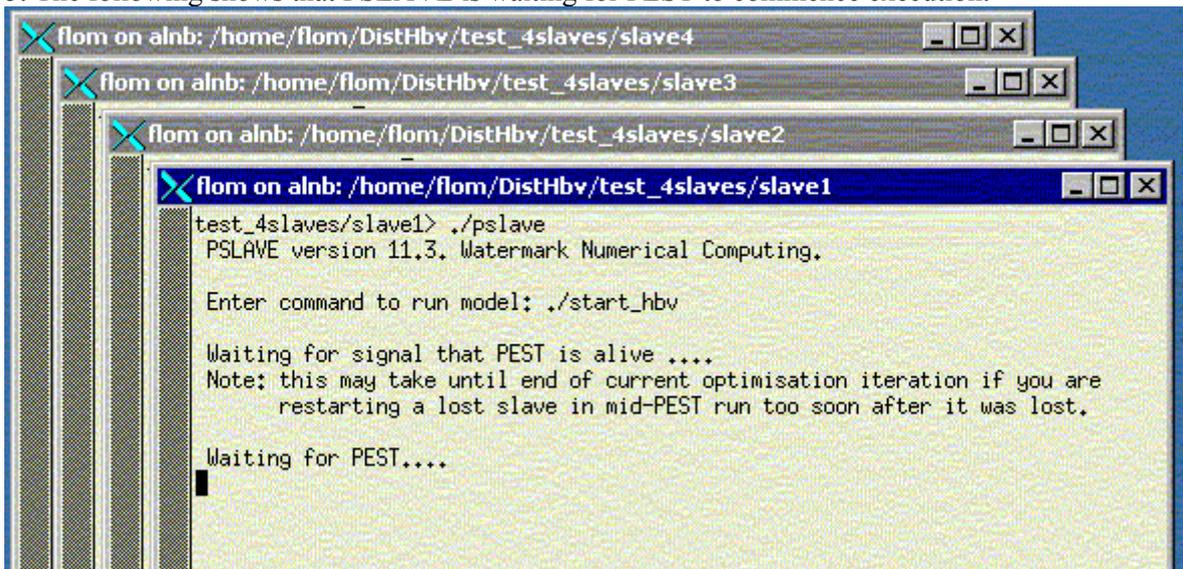
3. Starting PSLAVE by typing “pslave” on one PSLAVE command-line window. PSLAVE immediately prompts the user for the command which it must use to run the model.



4. Typing in the “./start_hbv”. This is the same command as that in command section of PEST control file.



5. The following shows that PSLAVE is waiting for PEST to commence execution.



6. Moving to PEST working directory and typing “ppest hbv” to start PEST, or using a full pathname to start PEST as following.

```
flo on alnb: /home/flo/DistHbv/test_4slaves
DistHbv/test_4slaves> /usr/local/pest/utpakket/ppest hbv
```

7. PEST firstly tries to communicate with slaves. When it detects that one slave is alive it begins to commerce execution.

```
flo on alnb: /home/flo/DistHbv/test_4slaves
DistHbv/test_4slaves> /usr/local/pest/utpakket/ppest hbv
Parallel PEST Version 11.3, Watermark Numerical Computing.

Testing whether slaves are alive .....

- slave "Apple" has been detected.

Only 1 slaves are alive.
PEST will now commence the parameter estimation process using only these
slaves and will use the others if/when they become available.

PEST is running in Parameter Estimation mode.

PEST run record: case hbv
(See file hbv.rec for full details.)

Model command line:
./start_hbv

RUNNING MODEL FOR FIRST TIME .....
```

8. Starting rest of the PSLAVE by repeating point 3 and point 4. After this has been done, on PEST command-line window it will be shown that the rest of the slaves have been detected.

```
flo on alnb: /home/flo/DistHbv/test_4slaves

- slave "Apple" has been detected.

Only 1 slaves are alive.
PEST will now commence the parameter estimation process using only these
slaves and will use the others if/when they become available.

PEST is running in Parameter Estimation mode.

PEST run record: case hbv
(See file hbv.rec for full details.)

Model command line:
./start_hbv

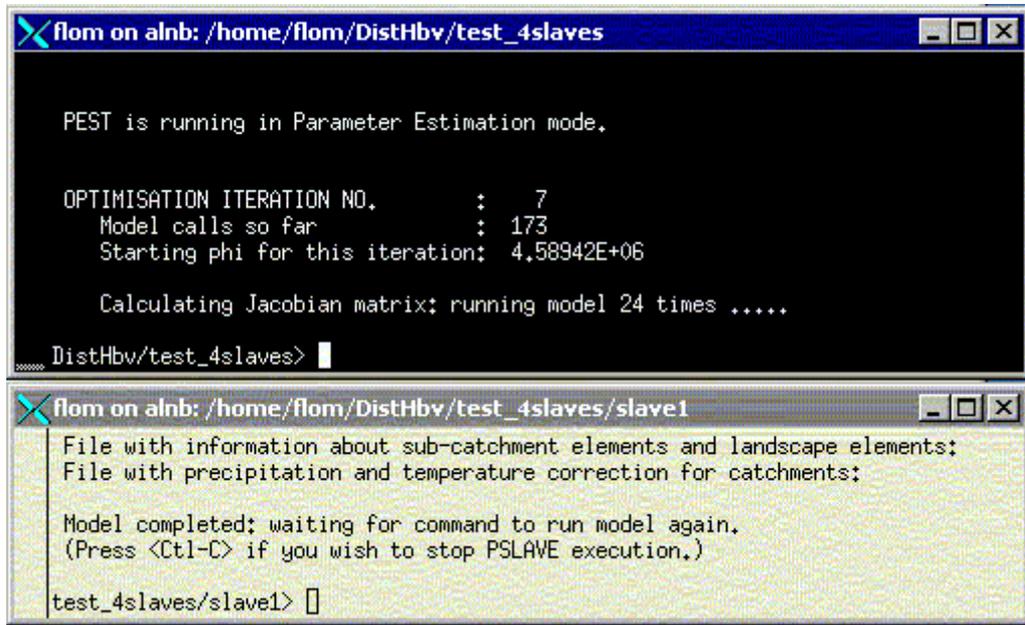
RUNNING MODEL FOR FIRST TIME .....
  Slave Banana has just been detected,

  Slave Cacao has just been detected,

  Slave Kiwi has just been detected,
```

Example 2. To restart Parallel PEST

1. The following image shows that a Parallel PEST execution was interrupted at iteration no.7.



```
fcom on alnb: /home/fcom/DistHbv/test_4slaves
PEST is running in Parameter Estimation mode.

OPTIMISATION ITERATION NO.      :    7
Model calls so far              :   173
Starting phi for this iteration: 4.58942E+06

Calculating Jacobian matrix; running model 24 times .....

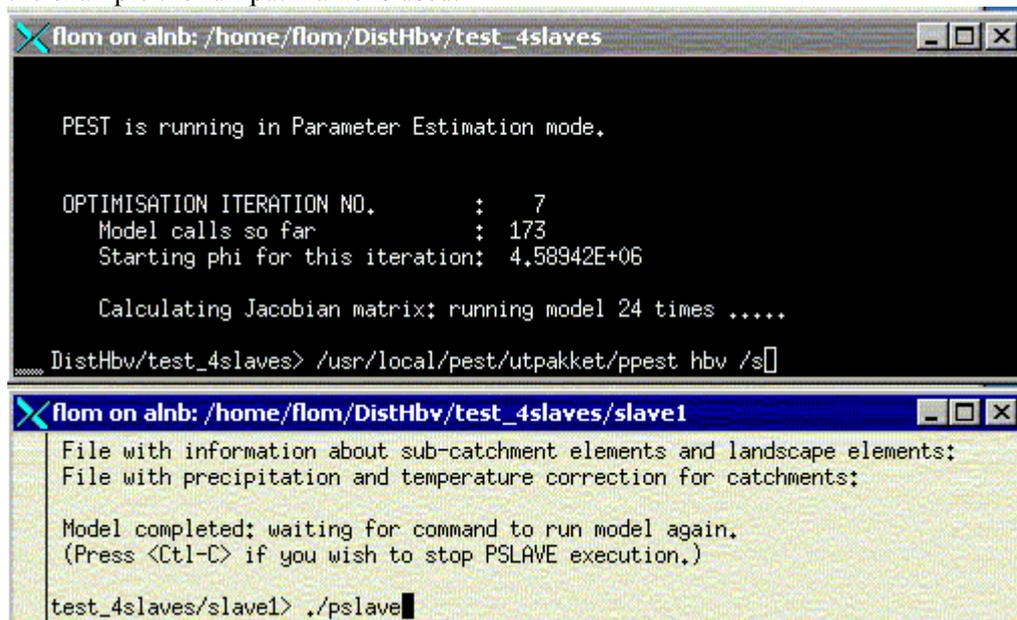
DistHbv/test_4slaves>
```

```
fcom on alnb: /home/fcom/DistHbv/test_4slaves/slave1
File with information about sub-catchment elements and landscape elements:
File with precipitation and temperature correction for catchments:

Model completed; waiting for command to run model again.
(Press <Ctl-C> if you wish to stop PSLAVE execution.)

test_4slaves/slave1>
```

2. The interrupted execution above can be restarted using “/s” switch. The slaves should be restarted before PEST. The command used to restart Parallel PEST is: ppest hbv /s. In the example the full pathname is used.



```
fcom on alnb: /home/fcom/DistHbv/test_4slaves
PEST is running in Parameter Estimation mode.

OPTIMISATION ITERATION NO.      :    7
Model calls so far              :   173
Starting phi for this iteration: 4.58942E+06

Calculating Jacobian matrix; running model 24 times .....

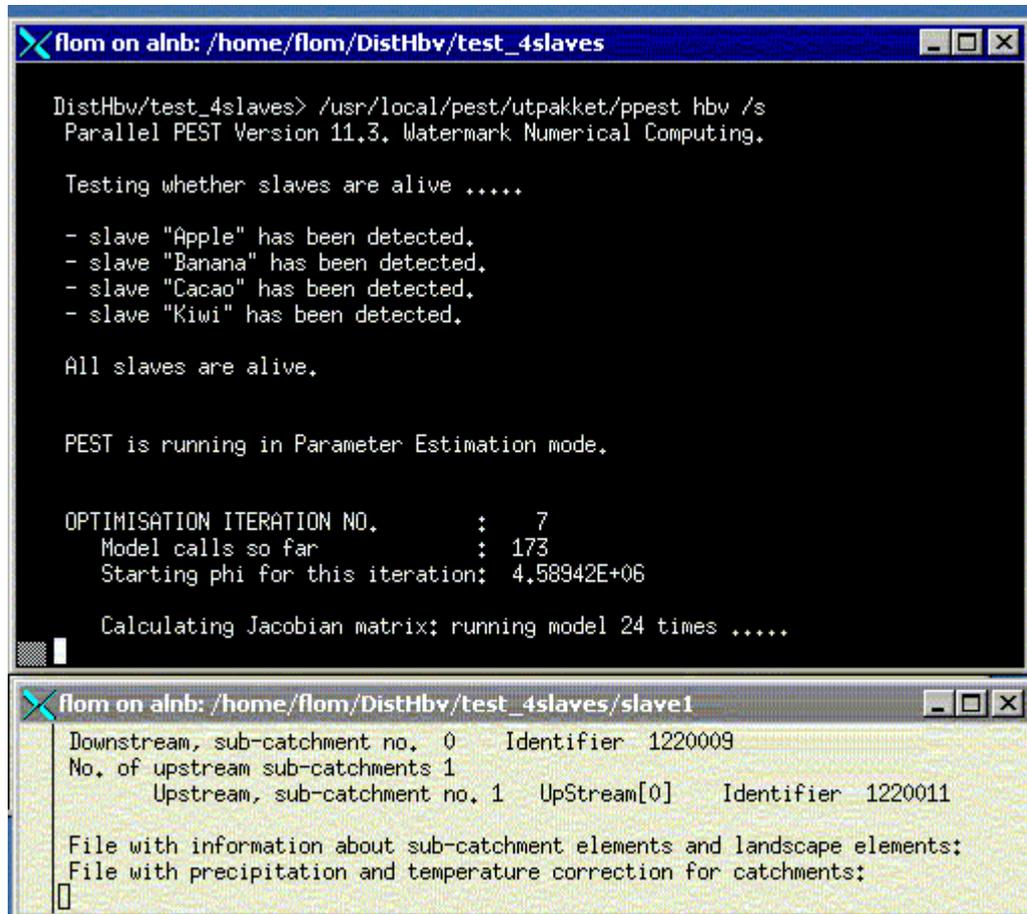
DistHbv/test_4slaves> /usr/local/pest/utpakket/ppest hbv /s
```

```
fcom on alnb: /home/fcom/DistHbv/test_4slaves/slave1
File with information about sub-catchment elements and landscape elements:
File with precipitation and temperature correction for catchments:

Model completed; waiting for command to run model again.
(Press <Ctl-C> if you wish to stop PSLAVE execution.)

test_4slaves/slave1> ./pslave
```

3. The following image shows that the execution of Parallel PEST is restarted from the iteration no. 7.



```
floam on alnb: /home/floam/DistHbv/test_4slaves
DistHbv/test_4slaves> /usr/local/pest/utpakket/ppest hbv /s
Parallel PEST Version 11.3, Watermark Numerical Computing.

Testing whether slaves are alive .....

- slave "Apple" has been detected.
- slave "Banana" has been detected.
- slave "Cacao" has been detected.
- slave "Kiwi" has been detected.

All slaves are alive.

PEST is running in Parameter Estimation mode.

OPTIMISATION ITERATION NO.      : 7
Model calls so far              : 173
Starting phi for this iteration: 4.58942E+06

Calculating Jacobian matrix: running model 24 times .....
```

```
floam on alnb: /home/floam/DistHbv/test_4slaves/slave1
Downstream, sub-catchment no. 0 Identifier 1220009
No. of upstream sub-catchments 1
Upstream, sub-catchment no. 1 UpStream[0] Identifier 1220011

File with information about sub-catchment elements and landscape elements:
File with precipitation and temperature correction for catchments:
```


This series is published by Norwegian Water Resources and Energy Directorate (NVE)

Published in the Report series 2008

No. 1 Gusong Ruan: Utilization of the new features in PEST (22 s.)