Calibrating a HSPF Model Using TSPROC and PEST

Note: it has been found that, depending on the operating system used by your machine, some of the results you obtain when doing this workshop may be a little different from those shown below. Do not be disturbed by this, as the differences should only be slight.

1. Introduction

1.1 About this Practical Exercise

In this exercise you will gain experience in using the comprehensive model post-processing functionality available through TSPROC in calibrating a HSPF model. However because TSPROC is not restricted in its use solely to calibration of HSPF, lessons learned through carrying out this exercise will be applicable to calibration and predictive analysis of many types of environmental models, particularly those that calculate and manipulate lengthy time series.

By the time you have finished this exercise you will have:

- used some of the many capabilities of TSPROC;
- undertaken parameter pre-processing using the PEST utility PAR2PAR;
- learned of the benefits to be gained through undertaking certain types of parameter transformation prior to parameter estimation;
- used PEST's predictive analyser to estimate the uncertainty associated with a specific model prediction;
- used TSPROC to estimate the parameters pertaining to a "sediment rating curve"; and
- incorporated sediment load observations into a parameter estimation process.

This exercise is based on "synthetic observations", ie. observations that were generated using the model itself (in this case HSPF). This has advantages and disadvantages. A disadvantage is that it would be more satisfying to actually solve a real-world problem, rather than a synthetic problem, in carrying out the work documented below. However an advantage is that the concepts illustrated in the discussion that follows can be explored without being clouded by issues related to failure of the model to represent a real-world environmental system. This is particularly important when it is recognised that many concepts need to be demonstrated in a short space of time; it would be difficult to demonstrate these concepts if more than a few parameters required estimation for a model that required calibration over a lengthy time period.

2. The Model

2.1 Description of the Model

Example.uci is a "User's Control Input" (ie. UCI) file for a simple HSPF model. As an inspection of this file reveals, this model is comprised of only one PERLND (pervious land segment) and one IMPLND (impervious land segment). The PERLND is far larger than the IMPLND, being 10000 acres in areal extent compared with only 800 acres for the IMPLND. Only the PWATER and SEDMNT sections of the PERLND are operative, while only the IWATER and SOLIDS sections of the IMPLND are active. Thus no precipitation is assumed to fall as snow. However erosion and transportation of solids within each land unit are simulated.

It is assumed that the PERLND and IMPLND drain to a common point, and that a gauging station which records daily flows exists at this point. Intermittent measurements of sediment load are also made at this point. Note however, that no in-stream flow and sediment processes are simulated by this simple model; hence no RCHRESs are included.

The simulation period covers 6 years, from the beginning of 1970 until the end of 1975. The model time step is one hour. Rainfall over the simulation time is shown in Figure 1.

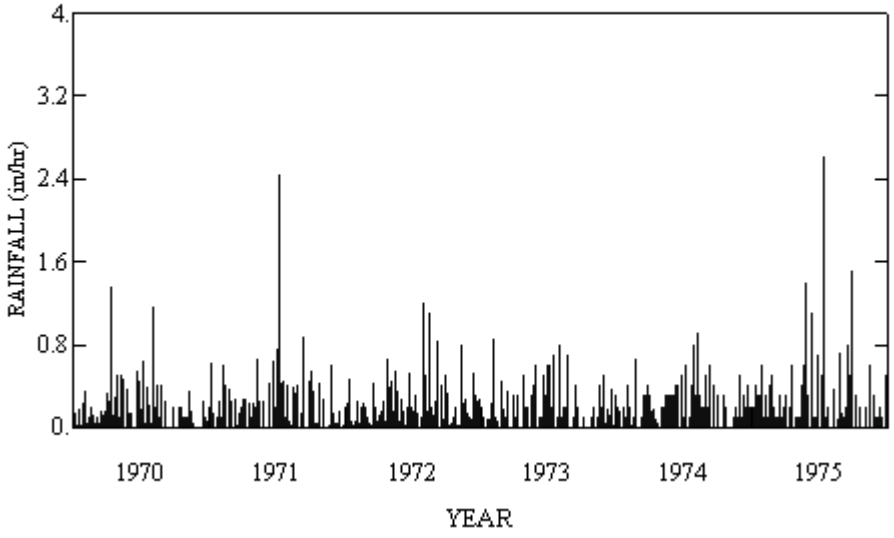


Figure 1. Rainfall over the model simulation time.

As can be seen through perusal of *example.uci*, HSPF is directed to read its rainfall and potential evaporation data from a Watershed Data Management (ie. WDM) file named *data.wdm*. Rainfall comprises Data Set Number 1 (ie. DSN 1) in this file, while potential evaporation comprises DSN 6 in this file. The latter is graphed in Figure 2.

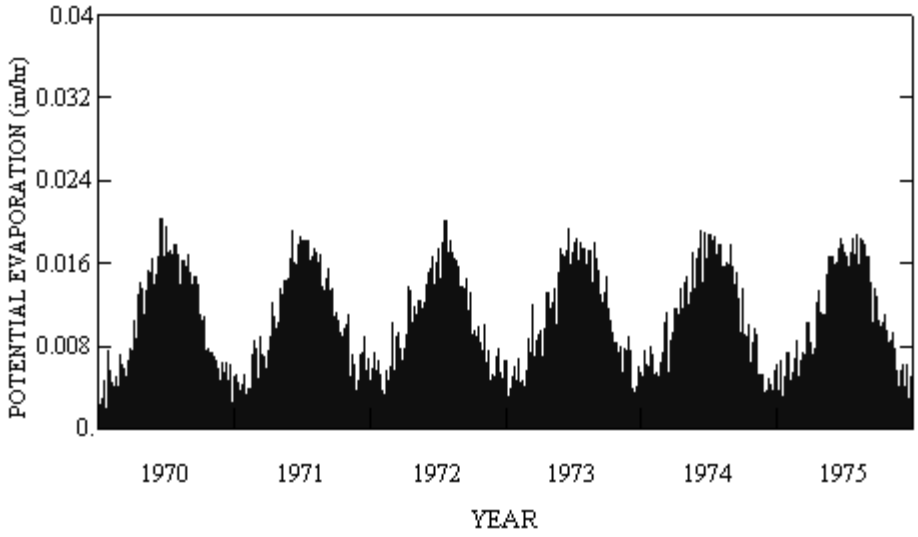


Figure 2 Potential evaporation over the model simulation time.

2.2 Observations

Daily flow observations are graphed in Figure 3. These are stored as DSN 8 in file *data.wdm*.

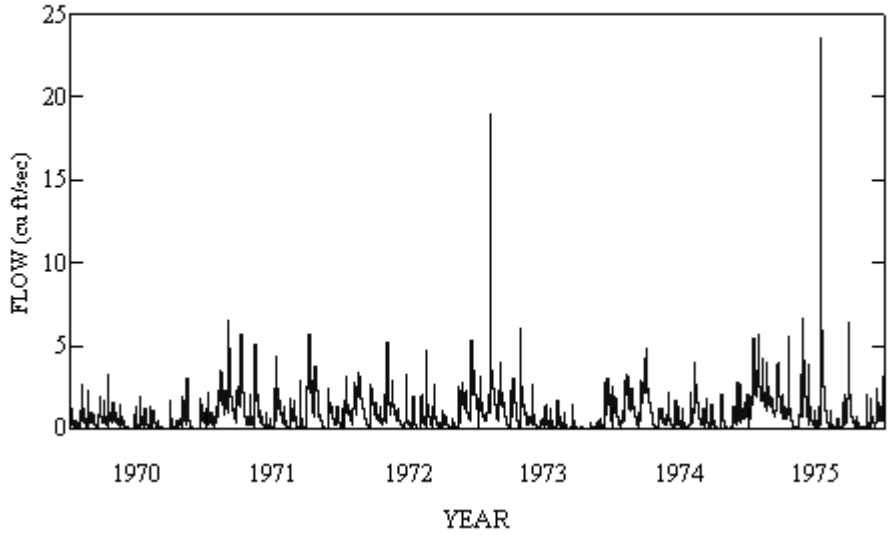


Figure 3. Daily flows measured over the model simulation time.

Sediment load observations over the model simulation time were made only intermittently. These are recorded in "site sample file" format in file *sed.smp*. The format for this type of ASCII file is described in the manual to the PEST surface water utilities; this format is also readily apparent from an inspection of file *sed.smp* using any text editor. Units for the sediment load observations recorded in file *sed.smp* are lb/ft^3. (It is acknowledged that these "observations" are a little unrealistic,

in that no account is taken of the transportation capacity of the stream, including the fact that its suspended load is highly dependent on its velocity. However the use of this synthetic dataset serves the present purpose of illustrating how to make best use of such data in the calibration process using a model that is not too complex and that is, by design, a perfect simulator of the "system" that was used to generate these "observations" in the first place.)

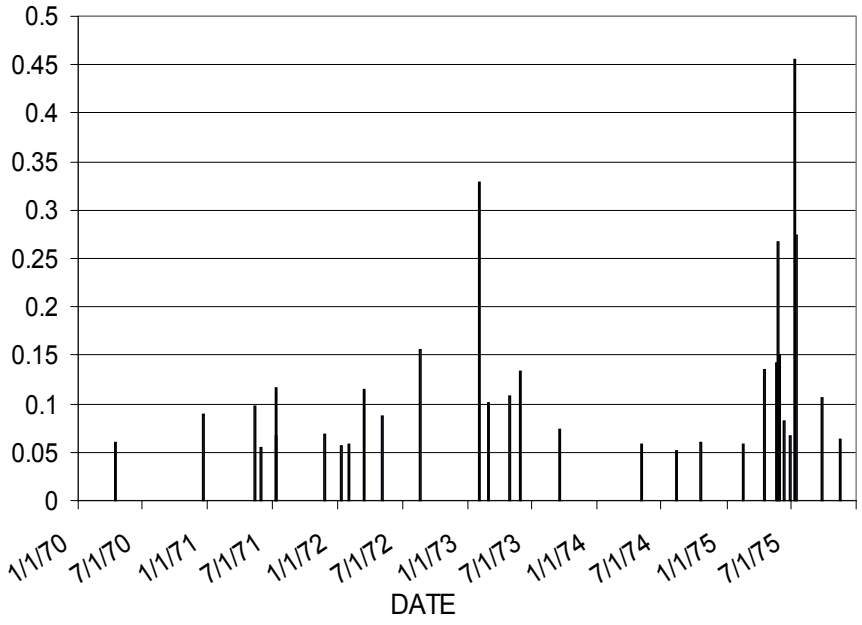


Figure 4. Intermittent sediment load measurements in lb/ft^3.

Sediment load "measurements" are graphed in Figure 4. The reader is reminded that the absence of graphed sediment load values between the measurement points depicted in Figure 4 does not imply zero sediment load. It only implies lack of measurements.

2.3 Running the Model

A special version of HSPF, named *xhspfx.exe*, is supplied for use in this exercise. This is a true WINDOWS executable; thus it is not prone to the problems that older DOS extended memory executables sometimes encountered when running under the WINDOWS operating system (especially WINDOWS NT). Moreover it runs without any screen display of the progress of the model run. This screen display is absent because the original version of HSPF which enabled such display in the command-line environment relied on calls to utility subroutines which are not compatible with the WINDOWS operating system.

To run the model, first open a command-line window (or a "DOS box" in the old terminology) and transfer to the working directory for the current exercise. Initiate execution of the model by typing the command:-

```
xhspfx example.uci
```

at the screen prompt. For the present model configuration, HSPF echoes its input data to file *example.ech*. Model outputs accessible through the PRINT-INFO sections of the PERLND and IMPLND blocks (which would have been written to file *example.out* in the present case) have not been requested. It is often wise to keep output to a minimum when running a model repeatedly under the control of PEST because such outputs will not normally be inspected, and valuable computing time may be lost in the writing of lengthy output files. Comprehensive model output can easily be reinstated by setting appropriate flags in *example.uci* if desired.

As is apparent from an inspection of file *example.uci*, model-generated flows and daily sediment loads will be recorded by HSPF in file *data.wdm* as DSN 1001 and DSN 1002 respectively.

The model can also be run by typing the command:-

```
xhspfx
```

at the command-line prompt. If run in this fashion, without a command-line argument, HSPF prompts for the name of its input file. Reply with:-

```
example.uci
```

and HSPF will run as above.

2.4 Modifications to HSPF

The version of HSPF used in this example has actually been enhanced from the "native" HSPF. These enhancements are such that HSPF can be directed to seek certain of its inputs from a file type other than its UCI file named a "supplementary input file". Numbers recorded in this supplementary input file can be supplied in free field format; thus they can be represented with the maximum number of significant digits allowed by a computer (this is often not possible on a UCI file). Where HSPF parameters are manipulated by a pre-processor (such as the PEST parameter pre-processor PAR2PAR – see later), it is important that numbers be transferred between the pre-processor and HSPF using maximum available precision, thus ensuring that accuracy in derivatives calculation is not degraded. (It should be noted, however, that when PEST writes numbers to a model input file itself this is not a problem, even where only a limited field width is available for the writing of a number. As is explained in the PEST manual, numerical imprecision is avoided by the fact that PEST adjusts its internal representation of a number to accord with that written to a model input file, even where the number is recorded with limited precision on this file.)

File *example0.uci* is slightly modified from file *example.uci* in that it makes use of the enhancements to HSPF discussed above. If you inspect this file with a text editor you will see that certain lines include the character strings "~*n*~" where *n* is an integer. The occurrence of such a string on a line of data read by the HSPF subroutine *rtable* (most data lines comprised of a set of real numbers are read by this subroutine) directs the modified HSPF to read this sequence of real numbers from a sequence with index *n* residing in a supplementary input file. The modified HSPF prompts for the name of

this file on the first occasion on which it encounters the *~n~* string. The supplementary input file which complements *example0.uci* is named *example0.sup*.

Run HSPF, informing it that its UCI file is *example0.uci*. When HSPF prompts for the name of a supplementary input file, reply with:-

```
example0.sup
```

HSPF should then run to completion.

Further details of the enhancements made to HSPF can be obtained from the documentation of this modified version.

3. TSPROC as a HSPF Post-Processor

3.1 TSPROC in a Nutshell

TSPROC stands for "Time Series PROCessor". Its primary purpose is to carry out calculations based on observed or model-generated time series. Such calculations include volumetric evaluation, basic statistical analysis, exceedence time calculation, and more. Because of these capabilities, it can function as a post-processor for models such as HSPF, "adding value" to time series generated by such models by enhancing the usefulness of their output datasets. However what makes TSPROC particularly useful as an aid to model calibration is this:- if a model-generated time series is to be compared with an observed time series, and/or various TSPROC-calculated functions of the former are to be compared with identical functions of the latter as part of the calibration process, TSPROC can generate an appropriate set of PEST input files by which model calibration can be undertaken based on these comparisons.

Another aspect of the design of TSPROC that makes it ideally suited for use in a calibration context is that time series manipulation carried out by TSPROC does not require that the time series involved in such manipulations have a constant sampling interval. Furthermore, as a fundamental part of the process of comparing observed time-series with their model-generated counterparts, TSPROC is able to interpolate the values of the latter to the sample dates and times of the former, thus ensuring that "apples are compared with apples" during the calibration process.

The results of TSPROC's calculations are stored as entities that reflect the types of data which they house. Thus time series are stored as "series", the outcomes of statistical calculations are stored as "s_tables", the outcomes of volumetric calculations are stored as "v_tables", and the outcomes of exceedence time calculations are stored as "e_tables". More such entities will be added over time as TSPROC's capabilities are expanded. Each entity must be provided with a unique name of 10 characters or less in length. The length restriction on the name reflects the fact that if such entities are used in a model calibration process undertaken by PEST, TSPROC uses these names to formulate observation and observation group names used by PEST (which are also of restricted length so as to conserve memory).

TSPROC receives instructions through a user-prepared input file. Within this file instructions are provided through a series of "blocks", each block containing the command to carry out some aspect of time series manipulation. As presently

programmed, time series can be imported into TSPROC from 3 file types, viz. a WDM file, a HSPF PLTGEN file, and a site sample file. As has already been discussed, the latter is an ASCII file type; in most cases it can easily be prepared by exporting time series data from an environmental database, supplemented possibly with a little extra manipulation using a standard text editor.

Each block in a TSPROC input file contains a set of keywords which provide TSPROC with the information it requires to carry out the processing requested by the block. The first keyword in any block must be the CONTEXT keyword. The instructions in the block will only be carried out if the block CONTEXT matches the TSPROC run CONTEXT (this being provided in the SETTINGS block which must lead any TSPROC input file) or if the CONTEXT for that block is supplied as "all". Through appropriate selection of the run CONTEXT, a single TSPROC input file can serve a variety of purposes. As will be shown below, this facilitates the use of TSPROC with PEST. Selection of one particular run CONTEXT setting will allow TSPROC to act as a model post-processor. Selection of another run CONTEXT setting in the same TSPROC input file will then allow it to be used in the generation of PEST input files for use in the calibration of a "composite model" comprised of the environmental simulator followed by TSPROC acting as a post-processor for that model.

Any line in a TSPROC input file beginning with the "#" character is ignored by TSPROC. Thus comment lines can be freely inserted in a TSPROC input file. This can be very useful in many situations.

3.2 Using TSPROC

File *tsproc0.dat* is a TSPROC input file. Inspect this file while reading this section.

The blocks found in file *tsproc0.dat*, and the instructions encapsulated in those blocks, are as follows:-

1. The SETTINGS block informs TSPROC that the current context is "model_run" and that the format used for date representation is *mm/dd/yyyy* (as opposed to *dd/mm/yyyy*). The CONTEXT is an arbitrary string supplied by the user. Subsequent blocks are processed only if their CONTEXT string matches that supplied in the SETTINGS block, or if their CONTEXT string is "all".

2. In the GET_SERIES_WDM block a time series is retrieved from DSN 1001 of the WDM file *data.wdm*. It is assigned to a series named *nflow*. As is apparent from an inspection of the HSPF input file *example0.uci*, this is the model-generated flow series.

3. Another flow time series is read from DSN 8 of the same file in the ensuing GET_SERIES_WDM block. As was mentioned above, this is the observed flow time series; it is assigned the name *oflow*.

4. In the NEW_TIME_BASE block, a new time series named *mflow* is generated by time-interpolation of the model-generated time series to the dates and times at which flow observations were made. Thus series *oflow* and *mflow* are now

directly comparable, as are any secondary quantities calculated in identical fashion from these series.

5. In the VOLUME_CALCULATION block monthly flow volumes are accumulated based on the *mflow* time series. Date intervals of volumetric accumulation (which all span periods of one month) are provided in file *dates.dat*. The v_table produced by this operation is named *mvol*.

6. Exceedence times for various flow thresholds are calculated using the EXCEEDENCE_TIME block. The e_table produced by this process is named *mtime*.

7. Using the LIST_OUTPUT block the *mflow* time series, together with the *mvol* v_table and the *mtime* e_table, are written to a file named *model.out*. Using the SERIES_FORMAT keyword, it is requested that dates and times be written to this output file along with series values themselves.

Run TSPROC using the command:-

```
tsproc
```

Answer its prompts as follows:-

```
Enter name of TSPROC input file: tsproc0.dat
Enter name for TSPROC run record file: tsproc.rec
```

As it runs TSPROC echoes the instructions contained within its input file to the screen, together with its response to these instructions. As this will normally scroll by too quickly to read, TSPROC records the same information to its run record file. If desired, this can be inspected after TSPROC has run to completion. TSPROC has comprehensive error-checking facilities. If any problems are encountered as TSPROC tries to process the blocks contained in its input file, it will cease execution with an appropriate error message, written both to the screen and to its run record file.

Inspect file *model.out*. This contains the results of TSPROC's calculations.

A "composite model" comprised of HSPF followed by TSPROC can be prepared by supplying the command to run HSPF followed by the command to run TSPROC in a batch file. Such a file is *model0.bat*. Inspect this file. Note that the command to run each of HSPF and TSPROC have been provided with responses to their screen prompts supplied through the keyboard input re-direction mechanism. This saves the user from having to answer these prompts as the composite model runs (this being essential when the composite model is run repeatedly by PEST). Keyboard input to HSPF is provided in file *xhspfx0.in* while keyboard input for TSPROC is provided in file *tsproc0.in*. These inputs are the same as those that were provided above when each of these programs was run "by hand". Inspect each of these files to verify that this is the case.

Now run the composite model by typing:-

```
model0
```

at the screen prompt. The screen display of each of the models cited in the batch file scrolls by as each of them runs in sequence. When such a composite model is run by PEST it is sometimes desirable that such model-generated screen output be disabled so that the model's screen output does not interfere with that of PEST. In the present instance, this is achieved using the batch file *model0a.bat* as the model. Inspect this file. Note the "@echo off" command at the top of this file, and the fact that all screen output from each sub-model is directed to a "nul" file (ie. to nowhere) using the ">" symbol. To verify that the model produces no screen output, type the command:-

```
model0a
```

at the screen prompt, and watch nothing apparently happen. To verify that something did, indeed, happen, obtain a directory of the files within the current working directory using the command:-

```
dir /od
```

(The "/od" qualifier means "in order of date".) The dates and times pertaining to the final files displayed in the list that is written to the screen as a result of the above command are proof that these files were, in fact, just written.

4. Parameter Estimation using PEST and TSPROC

4.1 Definition of Adjustable Parameters

As is documented in the PEST manual, adjustable parameters are identified on a model input file by preparing a template of this file. A template file serves two purposes. First, it informs PEST of the locations of adjustable parameters on a model input file; second, it provides names for these parameters. Each such parameter name must be unique and of 12 characters or less in length.

File *example1.tpl* is a template file built from a HSPF UCI file. Inspect this file. The first line of a template file always contains the string "ptf" followed by the "parameter delimiter" for that file. This is a (user-specified) character that is used to bracket parameter spaces in a template file (ie. the spaces to which PEST can write current values for nominated parameters). Within the body of the template file the pertinent parameter name is written between a pair of such delimeters, the delimiters and the name collectively identifying the space to which PEST must write the current value of the nominated parameter on each occasion that it runs the model. In the present instance it is very important that such parameter spaces respect the strict formatting requirements of a HSPF input file.

By searching for the "$" character, you will soon verify that four parameters are identified for estimation in *example1.tpl*. These are *lzsn* (lower zone nominal storage), *infilt* (index to infiltration capacity), *uzsn* (upper zone nominal storage) and *nsur* (Manning's *n* for the overland flow plane).

4.2 Using TSPROC to Prepare for a PEST Run

As has already been discussed, TSPROC can be used both as a model post-processor and as a PEST pre-processor. When used in the latter capacity it generates the input

files (except for model template files) required for a PEST run. Such input files are generated for calibration of a composite model comprised of the simulator (in this case HSPF) followed by TSPROC (acting as a model post-processor). Included in the calibration process can be any quantities calculated by TSPROC on the basis of observed and corresponding model time series. This brings tremendous power to the inversion process, for it means that model parameters can be adjusted such that the model tries to replicate not just historically observed flows, but quantities such as volumes and exceedence times based on these flows as well. Inclusion of such "value-added observations" can be vital to the success of the calibration process, for it is a general rule that if a model is required to predict a certain quantity in the future, then it should be calibrated at least partially against historical "observations" of that same quantity. Thus, for example, if a model is being built to predict exceedence probabilities under future climatic or management scenarios, then model-generated exceedence times (calculated *after* model-generated flows have been interpolated to the times of observed flows) should be compared with exceedence times calculated from observed flows. This is a *necessary*, but far from *sufficient*, condition that the model's performance in predicting future exceedence times will be acceptable.

Inspect file *tsproc1.dat*, a TSPROC input file. This is very similar to file *tsproc0.dat*; however certain blocks have been added to this file. Also, another CONTEXT keyword has been added to the SETTINGS block while the previous "model_run" CONTEXT string has been commented out. Using the new "pest_prep" context string, the additional processing blocks which differentiate file *tsproc1.dat* from the previous *tsproc0.dat* TSPROC input file are activated. These new blocks are used to build a PEST input dataset for a composite model consisting of HSPF and TSPROC, with the latter acting as a post-processor, implementing the instructions supplied in *tsproc0.dat*. But if the "pest_prep" CONTEXT string is commented out and the "model_run" CONTEXT string is uncommented, then TSPROC's actions under the control of *tsproc1.dat* will be identical to its actions under the control of *tsproc0.dat*; this is how it will be used as a model post-processor in the upcoming calibration run.

Tasks carried out by TSPROC under the control of *tsproc1.dat* (with the "pest_prep" context selected) which are additional to those carried out under the control of *tsproc0.dat* are as follows:-

1. Monthly flow volumes are calculated on the basis of the observation time series using the dates and times listed in file *dates.dat*, ie. the same dates and times that are used for calculation of volumes based on the modelled time series.

2. Exceedence times are calculated for the observation time series using the same flow thresholds that are used for exceedence time calculations based on the modelled time series.

3. PEST input files are prepared using the WRITE_PEST_FILES block.

If it appears within a TSPROC input file, a WRITE_PEST_FILES block must immediately follow a LIST_OUTPUT block in which a "model output file" is written by TSPROC in its capacity as a model post-processor. The PEST input dataset generated by the WRITE_PEST_FILES block is written under the assumption that TSPROC will be run as part of a composite model to be calibrated by PEST, and that

the principal output file of this model will be that cited in the immediately preceding LIST_OUTPUT block.

Tasks carried out by TSPROC in processing a WRITE_PEST_FILES block are as follows:-

1. TSPROC reads the template file cited in the WRITE_PEST_FILES block (in this case *example1.tpl*) in order to acquire the names of the parameters involved in the forthcoming parameter estimation run. This template file can optionally be linked to a model input file (in this case *example1.uci*). If it is not so linked, TSPROC will supply a dummy model input file name when it generates the PEST control file; the user must then alter this name by direct editing of the PEST control file before PEST is actually run.

2. If the PARAMETER_DATA_FILE and PARAMETER_GROUP_FILE keywords are present within a WRITE_PEST_FILES block, TSPROC reads these files in order to obtain appropriate parameter and parameter group settings for the PEST control file which it must generate. If these files are not supplied, TSPROC provides default settings for all pertinent variables.

3. TSPROC writes instructions to read the TSPROC-generated composite model output file named in the preceding LIST_OUTPUT block (in this case *model.out*). In doing this, TSPROC collects the elements of each entity cited in this block (ie. each time series, v_table, e_table and s_table) into a group of its own named after the entity. The name of the PEST instruction file to which these instructions are written must be provided with the NEW_INSTRUCTION_FILE keyword.

4. For each entity written to the model output file, a corresponding entity must be provided as "observation data". Elements belonging to the latter entities are written to the "observation data" section of the PEST control file as calibration targets. Weights are assigned to individual observations according to a user-supplied formula of arbitrary complexity provided with the WEIGHTS_EQUATION keyword. For time series entities, this equation can involve any time series with the same time base as the observation time series, certain intrinsic TSPROC functions such as time of year (we will encounter this later), and the value of the particular observation to which a weight is being assigned.

The last point is worth further explanation. TSPROC uses the *@_abs_value* function to extract the value of a particular observation for weights assignment purposes; this function can be used in conjunction with all TSPROC entities. Its use is exemplified in file *tsproc1.dat* following the SERIES_WEIGHTS_EQUATION keyword. In this equation, weights for individual flows are calculated as the reciprocal of observed flows; to avoid a numerical error if any flows are zero, a small number is added to each flow, 0.001 in the present case.

To generate a PEST input dataset for the present parameter estimation problem, run TSPROC using the command:-

```
tsproc
```

Respond to its prompts as follows:-

```
Enter name of TSPROC input file: tsproc1.dat
Enter name for TSPROC run record file: tsproc.rec
```

As it runs, TSPROC carries out all of the model post-processing tasks previously undertaken when it was run under the control of the TSPROC input file *tsproc0.dat*. However on this occasion it also generates a complete PEST input dataset comprised of a PEST control file named *case1.pst* and an instruction file named *model.ins*. As is explained above, the template file *example1.tpl* had already been prepared.

Check that the entire PEST input dataset is consistent and correct using PESTCHEK by typing the command:-

```
pestchek case1
```

PESTCHEK should report no errors or inconsistencies.

4.3 Final Tasks before Running PEST

As has been discussed, PEST will be asked to calibrate a composite model comprised of HSPF followed by TSPROC. As is indicated following the MODEL_COMMAND_LINE keyword in the WRITE_PEST_FILES block of *tsproc1.dat*, the individual executable programs comprising this model will be run through the batch file *model1.bat*. An inspection of this file (which must be supplied by the user) reveals that it will first run HSPF based on the UCI file *example1.uci* (of which *example1.tpl* is the template). Then it will run TSPROC, directing it to file *tsproc1.in* for its keyboard input.

Inspect file *tsproc1.in*. The contents of this file direct TSPROC to read the TSPROC input file *tsproc1.dat*, and to record its run record in file *tsproc.rec*. However there are two further entries in this file. As you can readily verify by running TSPROC again on the basis of file *tsproc1.dat*, TSPROC will inquire from the user whether it is alright to overwrite a previously-written PEST instruction file and a previously-written PEST control file if the instructions in a WRITE_PEST_FILES block direct it to do so. When TSPROC runs as a model post-processor, the CONTEXT setting should be changed so that this does not happen (because the WRITE_PEST_FILES block should be de-activated). However if you forget to alter the CONTEXT setting, use of the two "n" characters in response to TSPROC's requests to overwrite these files will prevent TSPROC from inadvertently destroying them (which would be a painful experience if you had edited them after TSPROC had originally generated them). Instead, TSPROC will immediately cease execution.

Before running TSPROC as a model post-processor under the control of PEST, edit file *tsproc1.dat*, commenting out the CONTEXT keyword that provides the run context as "pest_prep" and uncommenting the CONTEXT keyword that provides the run context as "model_run". After you have completed this task, the SETTINGS block should appear as below.

```
START SETTINGS
  DATE_FORMAT mm/dd/yyyy
  CONTEXT model_run
#  CONTEXT pest_prep
```

```
END SETTINGS
```

4.4 Running PEST

Run PEST using the command:-

```
pest case1
```

Note how PEST writes to the screen the contribution made to the objective function by each observation group. In practice, you may have to use this information to adjust the weights equations supplied in the WRITE_PEST_FILES block of the TSPROC input file until each observation group (based on each TSPROC entity) contributes roughly the same amount to the objective function.

There is no need to let PEST run to completion if time is short. If you wish to halt its execution, open another command-line window, transfer to the current working directory in that window, and type the command:-

```
pstopst
```

PEST will then cease execution with a full statistical printout on its run record file. (It will also provide a full statistical printout if allowed to run to completion.) An inspection of these statistics (particularly the covariance, correlation coefficient and eigenvector matrices) reveals that no parameter is particularly highly correlated with any other parameter. However parameter *nsur* is less sensitive than any other parameter, an impression reinforced by inspection of the sensitivity file *case1.sen*.

The fit between observed and modelled data is very good. Figure 5 shows the observed and model-generated flow time series plotted together for the year 1971. Observed and monthly flow volumes over the whole simulation time are plotted in Figure 6, while observed and monthly exceedence fractions are plotted in Figure 7. Indeed, there is every reason to be pleased with the fit that PEST has achieved (and with the ease with which the PEST input dataset for this complex calibration process was constructed).

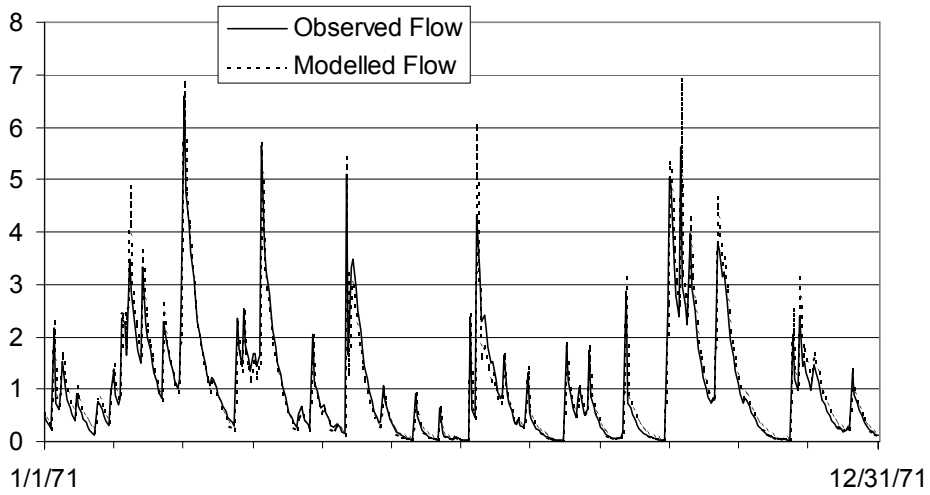


Figure 5. Modelled and observed flows (in ft^3/sec) during 1971.

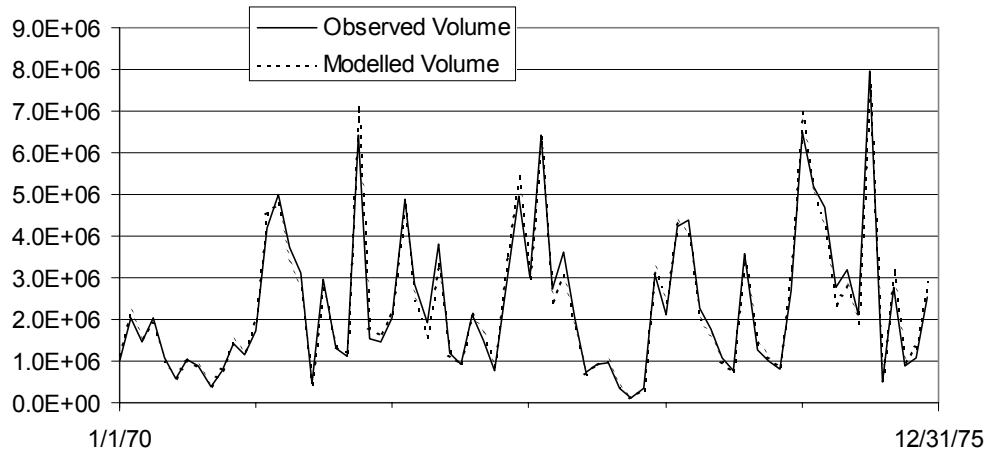


Figure 6. Modelled and observed monthly volumes (in ft^3) over the simulation time.

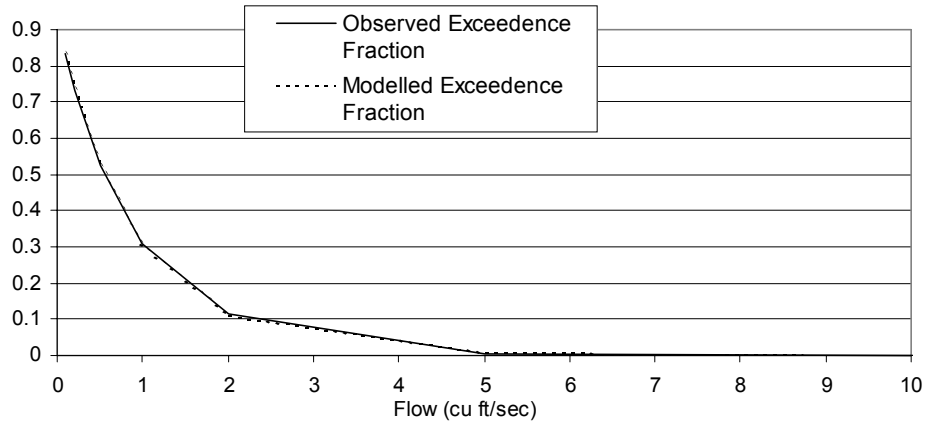


Figure 7. Modelled and observed exceedence fractions for the simulation time.

5. Parameter Pre-Processing

5.1 Supplementary Input File

Inspect file *example2.uci*. Data supplied to the PWATER section of the PERLND block is slightly different in this file to that which was supplied in file *example1.uci*; in *example2.uci* the upper zone nominal storage volume (UZSN) and Manning's *n* for the overland flow plane (NSUR) are now supplied at monthly intervals.

We would like to incorporate this seasonal dependence of UZSN and NSUR into the parameter estimation process. However, before we can do this, there are some

problems that need to be overcome. The first problem is that if we were to attempt to estimate each of these monthly quantities separately, this would bring 24 new parameters into the parameter estimation process. Furthermore, estimation of each of these monthly quantities on an individual basis ignores the fact that they are not really independent, and that we actually know something about the nature of the seasonal variation which they undergo.

The second problem is that HSPF allows numbers which represent these monthly quantities to be written to its input file with only limited precision. Thus, for reasons mentioned above, if any pre-processing is undertaken on these parameters, the loss of precision involved in recording the processed numbers to the HSPF input file will result in serious loss of accuracy in the calculation of model output derivatives with respect to these parameters.

The first of these problems will be overcome by assuming that each monthly UZSN and NSUR value has a sinusoidal dependence on time of year (with an appropriate phase term to ensure that the peak occurs during the appropriate season). PEST will thus be asked to estimate the mean value and amplitude of variation of each of these quantities (actually a slight variation of this – see below); calculation of monthly values from these HSPF parameters will be undertaken by the PEST parameter pre-processor PAR2PAR. PAR2PAR will also write these monthly quantities to the HSPF input file.

The second of the above problems will be solved by using a supplementary HSPF input file. As was discussed in Section 2.4, data can be recorded in such a file using free field format with no restrictions on the number of significant digits used to represent each number. The occurrence of "~*n*~" strings (where *n* is an integer) in file *example2.uci* directs the modified HSPF's attention to such a supplementary input file. For the present example this supplementary input file is named *example2.sup*. It will be noticed that HSPF is directed to this file not just to read monthly UZSN and NSUR input parameters, but also to read values for the FOREST, LZSN, INFILT, LSUR, SLSUR, KVARY and AGWRC parameters (all of these parameters are supplied on the same line of a UCI file). As was done in the previous example, PEST will estimate values for the LZSN and INFILT parameters. With all adjustable parameters thus residing on the one file (viz. *example2.sup*), there will no longer be any need to provide a template file for the UCI file. Instead, for this parameter estimation run, a template file must be made for the HSPF supplementary input file. File *example2.tpl* is such a template file. Note that this template file contains the names of 26 parameters, viz. *lzsn, infilt, uzsn[1-12] and nsur[1-12]*.

5.2 Program PAR2PAR

PAR2PAR is a parameter pre-processor supplied with PEST. In the present case it is PAR2PAR, not PEST, which will write the HSPF supplementary input file *example2.sup* on the basis of the template file *example2.tpl* just before each model run is undertaken as part of the parameter estimation process. Hence PAR2PAR will need to be included in the composite model run by PEST. The command to run PAR2PAR will be placed in the model batch file comprising the composite model prior to the command to run HSPF.

In the upcoming calibration process, PEST will adjust only 6 parameters, viz. *lzsn*, *infilt* and two parameters governing the seasonal distribution of each of UZSN and NSUR. It is the role of PAR2PAR to transform the latter four parameters into 24 parameters which are the monthly values of UZSN and NSUR. The PAR2PAR input file by which this is achieved is file *par2par2.dat*. Inspect this file.

PAR2PAR is able to calculate new parameters using relationships of arbitrary complexity involving parameters whose values are already defined. It is then able to record any or all of the parameter values which it calculates to one of more model input files; like PEST, PAR2PAR writes model input files using template files built from those model input files. In the present instance, PAR2PAR will write file *example2.sup* on the basis of the template file *example2.tpl*. PAR2PAR is instructed to do this in the "template and model input files" section of its input file (near the bottom of the file). Note that while many different parameters might be defined and calculated in a PAR2PAR input file as part of the complex processing tasks undertaken by this program, only those parameters whose names appear in template files cited in the PAR2PAR input file will actually be recorded on a model input file by PAR2PAR prior to a model run.

Parameter values and relationships between parameter values are defined in the "parameter data" section of a PAR2PAR input file. An inspection of this section reveals the manner in which monthly UZSN and NSUR values are calculated using the parameters estimated by PEST. Each such monthly value is calculated using the average value for this quantity and the amplitude of variation of this quantity about its average value. An "offset" or "phase" of 270 days is employed in the argument to the *sin* function; the factor of 0.01721 appearing in the argument to the *sin* function is actually $2\pi/365.25$; this fulfils the requirement that the argument to the *sin* function must be supplied in radians.

As mentioned above, PAR2PAR will be run by PEST as part of a composite model prior to running HSPF. Instead of supplying parameter values to HSPF, PEST will actually supply parameter values to PAR2PAR. Hence a template must be constructed for the PAR2PAR input file *par2par2.dat*. Such a template file is *par2par2.tpl*.

A subtle, but important, measure is taken in order to avoid potential disaster in the calculation of seasonal parameter values. If PEST were asked to estimate the mean and amplitude of each of these seasonal variations, it is not impossible that it would, at some stage during the parameter estimation process, test an amplitude value that was greater than the mean. This would result in negative HSPF parameter values for some seasons, a situation which HSPF would react to in a very negative way – abandoning further processing with a terse error message. To avoid this possibility, together with the yearly mean of each of these seasonal parameters, PEST is actually asked to estimate the "normalised" or "fractional" amplitude, ie. the amplitude divided by the mean. By placing an upper bound of 1.0 on this latter parameter, the possibility of negative seasonal transgressions is entirely removed.

5.3 The TSPROC Input File

The TSPROC input file for the present case is *tsproc2.dat*. This is very similar to *tsproc1.dat* which was employed for the previous parameter estimation run. The only differences between these two files are in the WRITE_PEST_FILES block. In

tsproc2.dat TSPROC is asked to write a PEST control file named *case2.pst* and to look for parameter names in the template file *par2par2.tpl*. Initial parameter values and parameter group variables are provided in files *initpar2.dat* and *groups2.dat*. Also, the command to run the model is now "model2.bat".

5.4 The Model Batch File

Inspect file *model2.bat*. As was done for *model1.bat*, all screen output for this "model" is disabled; this is not essential, but it helps in monitoring the progress of the parameter estimation process if PEST and the model share the same screen window.

As was discussed above, program PAR2PAR is run prior to HSPF. It is directed to seek its input from file *par2par2.dat*. (The name of the PAR2PAR input file is supplied directly on the PAR2PAR command line, so there is no need to use the "<" character for keyboard input re-direction.)

HSPF is directed to read file *xhspfx2.in* to find the input data that it would normally receive from the keyboard. The data within this file tells HSPF to read file *example2.uci* as its User's Control Input file, and *example2.sup* as its supplementary input file.

5.5 Preparing the PEST Input Dataset

Run TSPROC, replying to its prompts as follows:-

```
Enter name of TSPROC input file: tsproc2.dat
Enter name for TSPROC run record file: tsproc.rec
File model.ins already exists. Overwrite it? [y/n]: y
```

Check the integrity of the PEST input dataset written by TSPROC by typing the command:-

```
pestchek case2
```

Ignore the two warnings; these pertain to the manner in which derivatives are calculated for the parameters which govern seasonal variation of NSUR and UZSN. In the interests of more robust derivatives calculation, derivative increments will be calculated using an absolute, rather than relative, increment for these parameters.

5.6 Running PEST

Before running PEST, alter the SETTINGS block in file *tsproc2.dat* so that the new run context is "model_run". After you make these alterations, the SETTINGS block should appear as below.

```
START SETTINGS
  DATE_FORMAT mm/dd/yyyy
  CONTEXT model_run
#  CONTEXT pest_prep
END SETTINGS
```

Then run PEST using the command:-

```
pest case2
```

If time is short, stop its execution (with statistics) when it appears that the objective function will fall no more.

The objective function falls to a very low value in this case because the synthetic flows used as the observation dataset were actually generated using sinusoidal variations of NSUR and UZSN. Hence the fit between model outputs and "field data" is nearly perfect. Furthermore, the parameter values estimated by PEST are almost exactly equal to those used to generate the synthetic data in the first place.

Inspect the matrices at the bottom of the run record file. It is apparent that the two NSUR parameters (ie. *nsurav* and *fnsurvar*) are relatively insensitive, and somewhat correlated. Their relative insensitivity is no surprise, for insensitivity of the NSUR parameter was apparent from the results of our previous calibration run.

6. Introducing a Recession Parameter

6.1 Preparing for the PEST Run

Another calibration exercise will now be performed. This is identical to the previous one except for the fact that one new parameter has been introduced to the parameter estimation process. This is the interflow recession parameter IRC, for which the PEST parameter name is *irc*. This parameter is cited on the template file to the PAR2PAR input file *par2par3.dat*, the template file being named *par2par3.tpl*. PAR2PAR undertakes no manipulation or transformation of this parameter; it simply "passes it on" to the HSPF supplementary input file, in this case named *example3.sup*.

The optional parameter data and parameter group files, whose names are provided in the WRITE_PEST_FILES block, have been updated to include the initial value, bounds and derivative increment for this new parameter. As can be seen from an inspection of file *initpar3.dat*, parameter *irc* is log-transformed, has an initial value of 0.3 and is provided with lower and upper bounds of 0.01 and 0.99 respectively.

Build the PEST input dataset for this new parameter estimation process by running TSPROC. Respond to its prompts in the manner shown below:-

```
Enter name of TSPROC input file: tsproc3.dat
Enter name for TSPROC run record file: tsproc.rec
File model.ins already exists. Overwrite it? [y/n]: y
```

Now check the entire PEST input dataset using the command:-

```
pestchek case3
```

Once again, ignore the warnings.

6.2 Running PEST

Edit file *tsproc3.dat*, altering the run CONTEXT in the SETTINGS block to "model_run" as was demonstrated above. Then run PEST using the command:-

```
pest case3
```

Let PEST run to completion; or halt its execution when the objective function appears to be falling no further. From monitoring PEST's screen output it is apparent that PEST has great difficulty in optimising the parameters. The objective function falls for a while, then rises, and then falls again. PEST "gets there" in the end (ie. lowers the objective function to almost zero), but it takes a tortuous and difficult route. Once again, estimated parameter values are equal to "true" parameter values. However one is left wondering whether it would have indeed "got there" had the calibration dataset not been a synthetic one.

Apparently introduction of the interflow recession parameter caused problems for PEST. In fact, this is often the case for HSPF recession parameters (AGWRC is another such troublesome parameter). The problem arises because of the way these parameters are used by HSPF; the relationship between these parameters and model outputs is, in fact, highly nonlinear.

6.3 Parameter Transformation

Fortunately the above problem is easily overcome through appropriate parameter transformation. If PEST is asked to estimate a *function* of a troublesome parameter, rather than the parameter itself, and if the relationship between the parameter *function* and model outputs is more linear than that between the parameter itself and model outputs, then there is a very good chance that PEST's performance will be greatly improved.

File *tsproc4.dat* is a TSPROC input file which is almost identical to *tsproc3.dat*. The only differences are in the WRITE_PEST_FILES block. Though differently named, most of the files cited in this block are identical to corresponding files used in the previous PEST run. However the template of the PAR2PAR input file for the current PEST run, ie. file *par2par4.tpl*, is different from its counterpart, *par2par3.tpl*, used in the previous run. Inspect *par2par4.tpl*. Here it is seen that PEST will actually be asked to estimate a parameter named *irctrans*; PAR2PAR then calculates the value of the native HSPF parameter *irc* from *irctrans* using the relationship:-

irc = irctrans/(1+irctrans)

It is not too difficult to show that if *irc* is to be maintained between values of 0.01 and 0.999 (as is required by HSPF), then *irctrans* can vary between 0.01 and 1000. These limits are provided for *irctrans* in the parameter data file *initpar4.dat*. Note also that the initial value provided for *irctrans* in this file (ie. 0.429) results in an identical initial value for *irc* (ie. 0.3) as that which was used in the previous PEST run. Hence any differences between PEST's performance in these two cases can be attributed to the transformation of *irc* made possible through the use of PAR2PAR.

Build the PEST input dataset by running TSPROC in the usual way, informing it that its input file is now *tsproc4.dat*. The resulting PEST control file is *case4.pst*. Check it, and the rest of the PEST dataset, with PESTCHEK.

6.4 Running PEST Again

Before running PEST, edit the TSPROC input file *tsproc4.dat*, altering the run context from "pest_prep" to "model_run". Then run PEST using the command:-

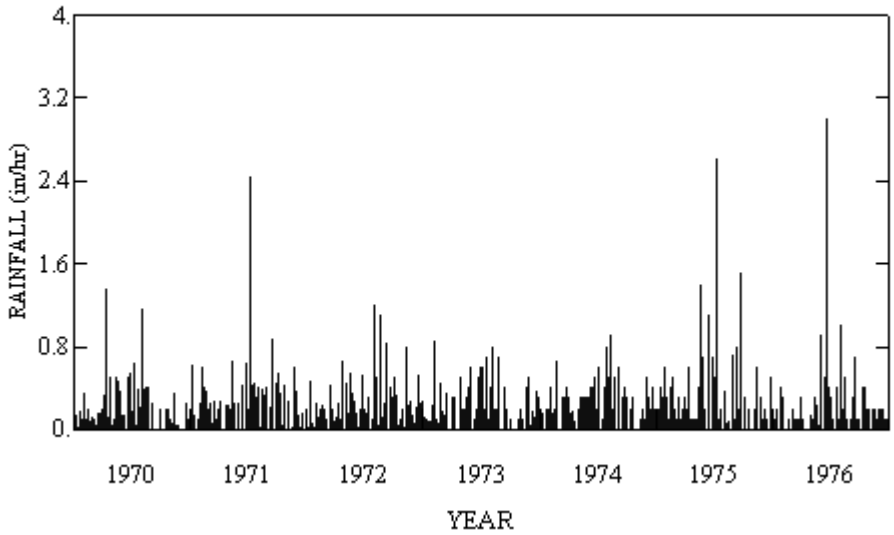
```
pest case4
```

The parameter estimation process is a little slow at first due to the fact that the *irctrans* parameter must change so much (as explained above, its potential range is far greater than that of *irc*), and that its change per iteration is limited by the factor change limit of 5 (PEST variable FACPARMAX) imposed on all factor-limited parameters. (This is a user-adjustable PEST variable; however a default value of 5 was supplied by TSPROC when it built the PEST input dataset.) Nevertheless, the objective function is rapidly lowered as the parameter estimation process progresses, and no signs of numerical instability are apparent. It is obvious that by estimating the transformed *irctrans* parameter, rather than the native HSPF *irc* parameter, the parameter estimation process becomes much more palatable to PEST due to its reduced nonlinearity.

7. Predictive Analysis

7.1 The Prediction

Figure 8 shows rainfall over the period 1970 to 1976; this period extends one year



beyond the calibration period, which ends at the end of 1975.

Figure 8. Rainfall over the calibration period, and for another year beyond that.

It is apparent from Figure 8 that although rainfall was, in general, lower in 1976 than it was in 1975 (the final year of the calibration period), there is a significant rainfall event in June of that year. We will now use the model to predict flow as a result of that rainfall event.

PEST's predictive analyser allows a model to be used in much more practical ways than has hitherto been possible. It is recognised that parameters determined through the process of model calibration are invariably non-unique. Nevertheless, parameters estimated through this process are constrained by the fact that the model must produce numbers which are acceptably close to their measured counterparts when the model is run under historical conditions. While the imposition of this constraint allows parameters to vary (sometimes to a shockingly large degree if they are highly correlated with each other or insensitive under calibration conditions), the nature of their allowed variation can be very complex. If parameters estimated through the calibration process are nonunique, a logical (and extremely important) question is "how does complex parameter nonuniqueness translate into nonuniqueness of predictions made by a calibrated model?"

PEST's predictive analyser can be used to explore the range of uncertainty of a key model prediction while maintaining the model in a calibrated (or almost calibrated) state. In doing this it makes no linearity assumptions concerning the relationships between model parameters and model outputs for, just like the nonlinear parameter estimation process undertaken by PEST, its predictive analysis process is a "nonlinear predictive analysis" process.

In the present case, predictive analysis will be based on the calibration run undertaken earlier in this exercise in which monthly variation of NSUR and UZSN was assumed to be nonexistent. If you had allowed the *case1* optimisation process to run to completion you would have found that PEST was able to lower the objective function to a value of about 218 by adjusting four parameters, viz. *nsur, lsur, uzsn* and *infilt*. As is apparent from Figures 5 to 7, this resulted in a very good fit between model outputs and field observations. Nevertheless it is possible that parameter values which are different from those actually estimated by PEST could also have resulted in fits which are this good, or nearly this good, over the calibration period. If this is the case, and if the predicted peak runoff rate during the June 1976 precipitation event is sensitive to these different parameter estimates, then the uncertainty associated with the prediction of peak runoff rate may be high. It is thus important that the uncertainty associated with this prediction be quantified.

7.2 The Input Dataset

A supplementary HSPF input file is not required for the present run. Furthermore, only four parameters will be varied. The UCI file for the present case is *example5.uci*. This differs from *example1.uci* only in the fact that the model is programmed to run until the end of 1976 instead of to the end of 1975. *Example5.tpl* is the template file for *example5.uci* in which the four adjustable parameters are identified.

A TSPROC input file named *tsproc5.dat* has been prepared. This differs from *tsproc1.dat* in that two SERIES_STATISTICS block have been added. In one of these blocks an s_table named "predict" stores the maximum flow calculated by the model over the period 6/19/1976 to 6/22/1976 (ie. the period of the rain event). Note that the maximum *daily* (rather than hourly) flow is calculated, even though rainfall is supplied to HSPF in hourly intervals, due to the averaging process undertaken by HSPF in forming its output flow time series. (As will be discussed below, a more spectacular result would have been calculated if such averaging had not taken place,

but that would have involved extra changes to the input dataset which would have then been a distraction from the principal aim of this exercise, which is to demonstrate the use of PEST's predictive analyser.) Another SERIES_STATISTICS block repeats this process, assigning the results of its calculation to the "opredict" s_table; this calculation is only undertaken when the run context is set to "pest_prep". It is done because when TSPROC generates a PEST input dataset, an observation must be available to match every model-generated quantity that features in the inversion process. When PEST runs in predictive analysis mode, the "observation" associated with the key model prediction which is being maximised or minimised serves no purpose; nor does the weight assigned to the observation. Nevertheless it must be present.

The LIST_OUTPUT block in file *tsproc5.dat* is also slightly modified from the corresponding block in file *tsproc1.dat* in that the s_table is now recorded on the model output file *model.out* together with other processed model outputs. The WRITE_PEST_FILES block links the model generated "predict" s_table to the "measured" "o_predict" s_table and provides a weight for this "observation". Because, as was stated above, this weight is ignored by PEST a weight of "1" suffices.

TSPROC can only write a PEST control file for cases where PEST runs in *parameter estimation* mode. It cannot write PEST control files for those cases where PEST runs in *predictive analysis* or *regularisation* modes. Fortunately it is an easy matter to adjust the former type of file so that PEST can run in either of these latter two modes.

7.3 Building the PEST Input Dataset

Before running TSPROC, HSPF must be run in order to generate a flow time series which is a year longer than those that it has been generating to date. If this is not done, it will not be possible to carry out the instructions contained within the SERIES_STATISTICS blocks of *tsproc5.dat* because these pertain to a period in June 1976, this being beyond the period of previous model runs. So run HSPF using the command:-

```
xhspfx example5.uci
```

Now build the PEST input dataset by running TSPROC, informing it that *tsproc5.dat* is its input file. After TSPROC has finished executing, open the PEST control file *case5.pst* which TSPROC has just written, and make the following changes:-

1. On the third line of file *case5.pst*, alter "estimation" to "prediction". This will inform PEST that it must run in *predictive analysis* mode.

2. Add the following lines to the end of the PEST control file.

```
* predictive analysis
1
300 320 500
0.00 0.005 1.0 2.0 8
0.00 0.05
4    0.0    0.005   4
```

The "1" under "* predictive analysis" in the above lines informs PEST that it must maximise (rather than minimise) the key model prediction. On the next line, PEST is informed that the model is deemed to be calibrated as long as the objective function is 300 or less (although 320 is acceptable); this objective function value still provides a very good fit between model outputs and field data. Thus PEST is able to adjust parameters *nsur, infilt, uzsn* and *lzsn* as much as it likes (while keeping them within user-supplied bounds) in order to maximise the prediction, as long as it ensures that the objective function is maintained at or below 320, but preferably as close to 300 as possible (recall that the optimised objective function was about 218). On the next line PEST is informed that it must refine its prediction maximisation process with a line-search procedure where applicable. Other variables contained in the "predictive analysis" section of the PEST control file pertain to termination criteria and modifications to the Marquardt lambda testing procedure (see the PEST manual for details).

The key prediction whose task it is for PEST to maximise or minimise is identified in the "observation data" section of the PEST control file as the sole member of the observation group "predict"; the "observed value" associated with this observation is ignored. The writing of an observation group named "predict" to the PEST control file by TSPROC was assured by giving this name to the pertinent s_table. As was mentioned above, TSPROC transfers entity names to observation group names when it generates a PEST input dataset.

Once you have made the changes indicated above, check your work with PESTCHEK by typing the command:-

```
pestchek case5
```

PESTCHEK should report no errors or inconsistencies. Now edit *tsproc5.dat*, altering the run CONTEXT from "pest_prep" to "model_run". After you have made this alteration, the SETTINGS block should appear as follows:-

```
START SETTINGS
  DATE_FORMAT mm/dd/yyyy
  CONTEXT model_run
#  CONTEXT pest_prep
END SETTINGS
```

6.4 Running PEST

Run PEST using the command:-

```
pest case5
```

Because initial parameter values for this run are the same as those that were used when PEST was earlier run in parameter estimation mode, the initial objective function is far higher than that at which we have deemed the model to be calibrated. So PEST runs in *parameter estimation* mode for a while, until it "sniffs" that the model is approaching a calibrated state. Then it works in *predictive analysis* mode, doing its best to raise or lower the key model prediction (ie. the maximum daily flow over the wet period in June 1976) while maintaining the model in a calibrated state.

Let PEST run to completion, or stop its execution when you feel as though the prediction will rise no higher. The maximised prediction is 124.76.

Now edit *case5.pst*, altering the "1" under the "* predictive analysis" line to "-1" (thus asking PEST to minimise the key model prediction rather than maximise it). Then run PEST again. The minimised prediction is 105.75. Hence the range of uncertainty pertaining to our key model prediction has been bracketed.

It is important that the reader be aware of the fact that the above range of predictive uncertainty (a little under 20% of the value of the prediction) is not very representative of the uncertainty associated with most predictions made by environmental models. The low range of predictive uncertainty in the present case is attributable to the following causes:-

1. The fit between model outputs and field data under calibration conditions is unusually good. The objective function at which we deemed the model to be uncalibrated also results in a very good fit between model outputs and field data.

2. Only four parameters were decreed as "adjustable"; thus parameter correlation is minimal.

3. The key model prediction whose task it was for PEST to maximise or minimise is the maximum *daily* flow over a user-defined period. The peak rainfall occurred at about 11am on the day of the major rainfall event. In adjusting the parameters at its disposal, PEST could maximise or minimise the total flow on the day of the event only through minimising or maximising the amount of residual flow taking place on the day *after* the event. This day begins sufficiently after the time of peak precipitation to make this procedure quite difficult for PEST. A different outcome would have probably occurred if PEST had been asked to maximise/minimise the peak *hourly* flow.

8. Using TSPROC to Calibrate a Sediment Rating Curve

7.1 Use of the Sediment Rating Curve in the Model Calibration Process

In this portion of the exercise we will use TSPROC not as a model post-processor, but as the model itself. A particularly powerful aspect of TSPROC's functionality is available through its SERIES_EQUATION block. Using this block, one time series can be calculated from one or a number of other time series with identical time base using an equation of arbitrary mathematical complexity.

In most cases where sediment or constituent measurements have been made in a stream, these measurements cannot be used directly in the calibration process. This is because the large amount of "noise" that is mostly associated with such data makes it very unlikely that model parameters can be adjusted in such a way that the model is able to come anywhere near calculating sediment or constituent concentrations that achieve a good fit with that data. Even if this were possible, it is likely that parameters would have to be "bent" to such unrealistic values to achieve such a fit that predictions made by the model using these parameters would be quite unbelievable.

On the other hand, sediment/constituent measurements should not be ignored either, for there may be valuable information contained within such a measurement dataset that will allow some adjustment of parameters such that the model is able to replicate certain aspects of the behaviour of the system, even if it cannot simulate the fine details of system behaviour. One possible mechanism for accommodating sediment and constituent data in the calibration process is to adjust model parameters such that the model has a similar rating curve to that which is observed. That is, parameters are adjusted such that the relationship between sediment load and stream flow magnitude that is observed in practice is respected by the model. The next section is devoted to incorporating the sediment rating curve into the parameter estimation process. The present section illustrates how TSPROC can be used in developing the sediment rating curve.

8.2 Displaying the Relationship between Sediment Load and Flow

The first step in developing a sediment rating curve is to plot observed sediment load against observed flow. As was discussed above, in the present case intermittent sediment load observations exist over the calibration period; see Figure 4. These observations are stored in site sample file format in file *sed.smp*. Flows at times corresponding to these sediment observations must now be obtained and written in a format where they can be read by a plotting and/or spreadsheet package together with the sediment observations to which they correspond. This can be easily achieved using TSPROC. A suitable TSPROC input file for this task is *sed1.tsp*.

In accordance with the instructions contained in file *sed1.tsp*, TSPROC first reads the site sample file *sed.smp*, extracting the sediment measurements contained therein. Then it reads observed flows from DSN 8 of the WDM file *data.wdm*. Next it generates a new time series comprised of flow observations time-interpolated to the dates and times at which sediment observations were made. This new time series, together with the observed sediment time series itself, are then written in ASCII format to file *sedflow.txt*. Run TSPROC using *sed1.tsp* as its input file. Then inspect file *sedflow.txt* to verify that both of these time series are recorded in it.

File *sedflow.txt* can easily be imported into a spreadsheet. If sediment load is plotted against flow, the graph shown in Figure 9 results.

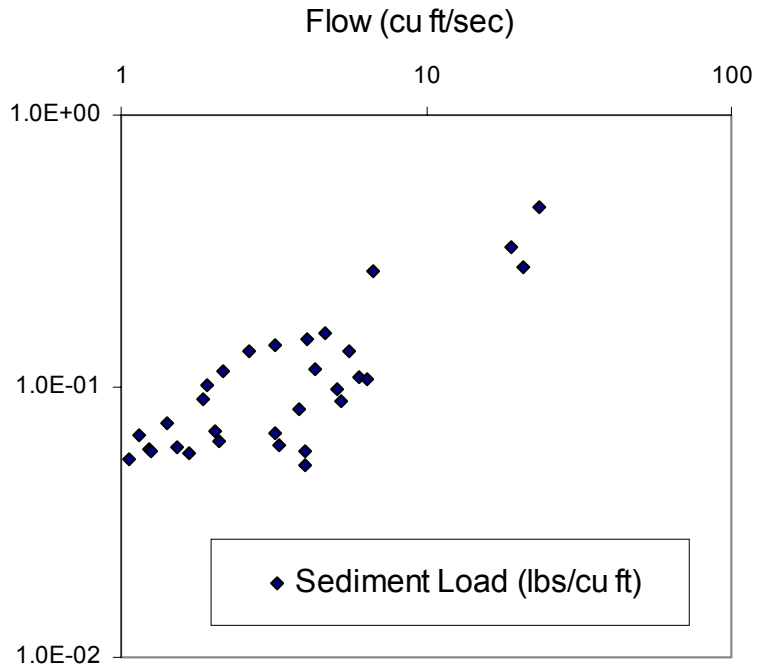


Figure 9. Observed sediment load plotted against observed flow.

8.3 Quantifying the Relationship between Sediment Flow and Load

Our next task is to fit a regression line to the data plotted in Figure 9. We will attempt to describe the relationship between sediment load and flow using the equation:-

$$\log(s) = a + b\,[\log(f)]^c + d\,\sin[(t + e)\,2\pi/365.25] \qquad (1)$$

where:-

s is the sediment load;

f is the flow;

t is the time in days since the beginning of the year; and

a, b, c, d and *e* are parameters to be determined through regression.

The TSPROC input file *sed2.tsp* can be used to calculate the log of sediment load from flow using equation (1). First measured load values are read into TSPROC so that measured flows can be time-interpolated to the dates and times at which these load measurements were made. Then "regression-calculated" sediment loads are generated at the sediment load measurement times using a SERIES_EQUATION block. The latter are then written to file *regsed.txt*. Note that in the equation provided in the SERIES_EQUATION block of file *sed2.tsp* it is assumed that *a* is 1.0, *b* is 0.1, *c* is 1.0, *d* is 1.0 and *e* is –100.0.

Run TSPROC, informing it that the name of its input file is *sed2.tsp*. Then inspect *regsed.txt*. It is obvious that the parameters that we have used in the regression equation are in need of some refinement, for the calculated logs of sediment load are nowhere near their "measured" values.

8.4 Using PEST to Calculate the Regression Parameters

As mentioned above, our "model" for this exercise will be TSPROC itself. The "model input file" will be *sed2.tsp*. Hence our first task is to build a template file from *sed2.tsp*; *sed2.tpl* is such a file. If you inspect this file you will find that parameter spaces have been inserted into the equation cited in the SERIES_EQUATION block in place of the parameter values cited above.

Our next task is to prepare another TSPROC input file, based on *sed2.tsp*, that will allow TSPROC to generate the PEST input files for the intended parameter estimation process. In all examples to date the TSPROC input file used in PEST file generation was the same as that used by TSPROC in its role as a model postprocessor; use of TSPROC for either of these two roles was selected using the run CONTEXT string. However in this case we will use different TSPROC input files for these two purposes, the reason for this being that it becomes a little cumbersome to perform both these roles with the same file when TSPROC itself is the model.

File *sed3.tsp* differs from *sed2.tsp* only through the fact that two extra blocks have been added. The first is a SERIES_EQUATION block by which the logs of measured sediment loads are calculated from the sediment loads themselves to form the "observation time series" to which the "modelled time series" (ie. the one produced by the regression equation) is matched. The second new block is the WRITE_PEST_FILES block. Using this block, TSPROC generates a PEST input dataset for the present parameter estimation task; the PEST control file is named *case6.pst*. The purpose of the parameter estimation process is to calculate a set of parameters (as defined in *sed2.tpl*) that allow TSPROC (run with *sed2.tsp* as its input file) to match regression-calculated logs of sediment loads with the logs of observed sediment loads as closely as possible.

In the WRITE_PEST_FILES block of *sed3.tsp*, TSPROC is instructed to write a PEST control file in which the model command line is "model6.bat". Inspect file *model6.bat*. In this file, TPROC is run using the command:-

```
trsproc < tsproc6.in > nul
```

Now inspect file *tsproc6.in*. Verify that TSPROC is provided with file *sed2.tsp* as its input file.

Before running TSPROC to build the PEST input dataset, inspect file *sedpar.dat*. This is the (optional) parameter data file for the present case. (It is optional because the PARAMETER_DATA_FILE keyword can be omitted from the WRITE_PEST_FILES block if desired. If this occurs TSPROC supplies default parameter data for the "parameter data" section of the PEST control file. Such data will nearly always require modification by the user before PEST is run.)

In file *sedpar.dat* two parameters are provided with non-zero OFFSET values. This is generally a good thing to do for parameters that can assume both positive and negative values. If such a parameter temporarily assumes a near-zero value in the course of the parameter estimation process, its capacity to change value as this process advances is severely limited by the RELPARMAX setting which governs the maximum relative change that any relative-limited parameter can undergo during any optimisation iteration. However, if provided with an appropriate OFFSET value, with offset upper and lower parameter bounds to match, such a parameter can be prevented from ever becoming zero, thus avoiding this problem altogether.

Inspection of file *sedpar.dat* reveals that, with OFFSET values taken into account, initial parameter values are the same as those used in the TSPROC run which was undertaken above.

To generate the PEST input files for the present parameter estimation problem, run TSPROC, supplying it with *sed3.tsp* as its input file. Then verify that the input dataset thus generated is consistent and correct by running PESTCHEK using the command:-

```
pestchek case6
```

Then run PEST using the command:-

```
pest case6
```

The objective function falls from an initial value of about 172 to a final value of about 0.54.

8.5 Processing the Results

After the parameter estimation process is complete, optimised parameter values can be found in the PEST run record file *case6.rec* and in the parameter value file *case6.par*. After accounting for the OFFSET supplied to parameters *a* and *e*, best-fit values for *a, b, c, d* and *e* are as recorded in Table 1.

Parameter	Optimised value
a	-1.1923
b	0.431162
c	1.84253
d	4.1057e-2
e	-32.555

Table 1. Optimised regression parameters for sediment rating curve.

In Figure 10 sediment load predictions calculated using the optimised regression equation are plotted against flow together with observed sediment load values. The fit between the two datasets is not bad.

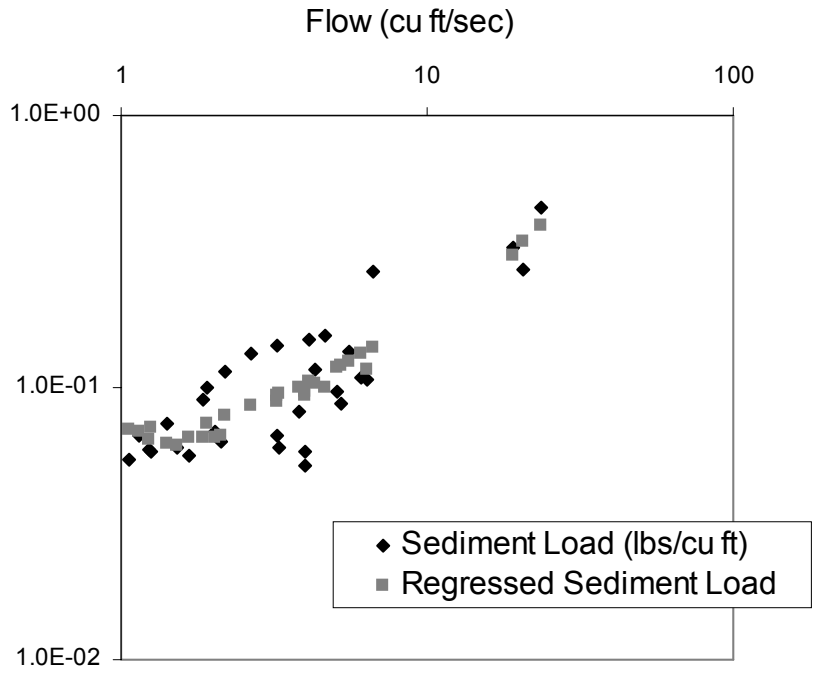


Figure 10. Measured sediment load and sediment load calculated from stream flow.

9. Incorporating Sediment Load Data into the HSPF Calibration Process

9.1 Strategy

As has already been discussed, it is often a pointless exercise trying to match model-generated sediment loads with measured sediment loads on a point-by-point basis. Instead, it is often a better strategy to adjust model parameters relating to erosion and sediment transport in such a way that certain characteristics of the model-generated sediment load distribution over time match the same characteristics of measured sediment loads. In the present case we will attempt to ensure that the sediment rating curve pertaining to model-generated sediment loads and flows matches the sediment rating curve based on observed loads and flows. It is important to note, however, that the "model-generated" sediment rating curve will be calculated using sediment load and flow values time-interpolated to the same dates and times as those at which sediment load measurements were made. In this way any seasonal, flow-magnitude, or temporal bias introduced into the sediment rating curve on account of the limited number of samples upon which its calculation was based, will be "cancelled out" as

the same bias is automatically incorporated into the calculation of the model-generated sediment rating curve.

The TSPROC input file for the present example is *tsproc7.dat*. This is based on *tsproc4.dat*. Nine parameters will be estimated; viz. *lzsn, infilt, irctrans* (the transformed interflow recession parameter), *uzsnav* and *fuzsnvar* (the mean and amplitude of monthly UZSN), *nsurav* and *fnsurvar* (the mean and amplitude of monthly NSUR) and the two sediment parameters *kger* and *jger*. The latter are the coefficient and exponent in the equation used by HSPF to calculate scour of the soil matrix (the dominant erosion mechanism in the present case). All of these parameters are featured in the template file *par2par7.tpl* of the PAR2PAR input file *par2par7.dat*. As in the preceding examples, PAR2PAR manipulates some of these parameters and "passes on" others (including the new scour parameters), before writing native HSPF parameters to a HSPF input supplementary file (in this case named *example7.sup*) on the basis of a template of this file (in this case named *example7.tpl*).

8.2 The TSPROC Input File

Inspect file *tsproc7.dat*. Additional blocks provided in this file to accommodate the inclusion of sediment data in the parameter estimation process are as follows:-

1. Observed sediment loads are read from the site sample file *sed.smp* using a GET_SERIES_SSF block.

2. Model-generated sediment loads (in tonnes per day) are read from DSN 1002 of the WDM file *data.wdm* using a GET_SERIES_WDM block.

3. Using a SERIES_EQUATION block, model-generated sediment loads are converted from tonnes/day to lbs/ft^3 by dividing by model-generated flows and multiplying by an appropriate conversion factor. Note how a small quantity is added to each term of the model-generated time series in this equation in order to avoid a divide-by-zero error.

4. Modelled sediment loads are then time-interpolated to the dates and times at which actual sediment load measurements were made. This is accomplished using a NEW_TIME_BASE block.

5. Next model-generated flows are time-interpolated to the dates and times at which sediment load measurements were made. These will be used to compute model-generated "regressed sediment loads", which will be compared with actual model-generated sediment loads.

6. Using a SERIES_CLEAN block, any flow terms less than 1.0 are increased to 1.0. If this is not done the log of the flow will be negative. When this negative term is then raised to a fractional power in the sediment rating curve equation a numerical error will result.

7. Using a SERIES_EQUATION block the logs of sediment loads calculated from model-generated flows using the optimised regression equation are subtracted from the logs of model-generated sediment loads. (Note the use of a small additive constant to terms of the *mosed* time series to avoid taking the log of zero, should such a zero-valued sediment load be generated by the model.)

8. If there was perfect agreement between model-generated sediment loads, and sediment loads estimated from model flows using the optimised sediment load regression equation, the terms of the series generated in the previous SERIES_EQUATION block would all be zero. Thus the "observed equivalent" of this model-generated time series is a time series comprised of all zero values. This time series is generated using a SERIES_EQUATION block in that part of the TSPROC input file devoted to preparation of the PEST input dataset for, unlike all of the operations described so far, this operation need only be carried out once.

9. The LIST_OUTPUT block which writes the "model output file" seen by PEST is updated such that the above-calculated "difference series" is also written to this file.

10. The WRITE_PEST_FILES block is updated to reflect the existence of the extra time series in the calibration dataset. As well as attempting to ensure that model-generated flows, volumes and exceedence times are well matched to their measured counterparts, PEST is now asked to ensure that the terms of the "difference time series" (ie. the difference between model-calculated sediment loads and model-generated regression-calculated sediment loads) are as close to zero as possible. A weight of 30 is assigned to each zero-valued "observation", this value having been determined by trial and error such that the contribution made to the overall objective function by this observation type neither dominates the objective function, nor is "overpowered" by the contributions made by other observation types to the objective function.

9.3 Preparing for a PEST Run

In common with most of the TSPROC input files with which we have been dealing so far, the TSPROC input file for the present example includes blocks that are processed when TSPROC is run both in a model post-processing capacity and in its capacity as a PEST pre-processor, as well as blocks that are only processed when TSPROC is run in the latter capacity. Selection of blocks for processing on any particular run is carried out using the run CONTEXT keyword in the SETTINGS block. As supplied, the run CONTEXT in *tsproc7.dat* is "pest_prep"; this setting is such that all blocks within the TSPROC input file are processed, including those used to generate the PEST input dataset.

On the previous occasion that HSPF was run, it was run over a 7 year simulation period; recall that the simulation time ran one year over the six year calibration period in order that predictive analysis could be carried out. The instructions in file *tsproc7.dat* were written under the assumption of a six year simulation time. So before running TSPROC to generate the PEST input dataset for the present case we

must ensure that all model results stored in file *data.wdm* are in accordance with the assumed run time for this case (ie. 6 years). So run HSPF using the command:

```
xhspfx
```

In order for it to read the UCI file for the current case, respond to its prompts as follows:-

```
Enter the name of your input file: example7.uci
Enter name of HSPF supplementary input file: example7.sup
```

Now run TSPROC, informing it that its input file is *tsproc7.dat*. TSPROC will, once again, generate a complete PEST input dataset; the pertinent PEST control file is *case7.pst*. Check this dataset using the command:-

```
pestchek case7
```

Ignore the warnings.

Next alter file *tsproc7.dat*, changing the run context from "pest_prep" to "model_run". After you have made these alterations the SETTINGS block should look like this:-

```
START SETTINGS
  DATE_FORMAT mm/dd/yyyy
  CONTEXT model_run
#  CONTEXT pest_prep
END SETTINGS
```

Then run PEST using the command:-

```
pest case7
```

9.4 PEST Run Results

An inspection of PEST's screen output and/or its run record file reveals that when the parameter estimation process is complete the biggest contributor to the objective function is that made by the observation group *seddiff*, ie. the group comprised of the differences between model-calculated sediment loads and sediment loads computed from the sediment rating regression equation on the basis of modelled flows. All other components of the objective function are quite low, indicating an excellent fit between model-generated and measured flows, volumes and exceedence times.

A comparison between observed and modelled sediment rating curves is provided in Figure 11. In this figure the dark diamonds perform the same role that they do in Figures 9 and 10; that is, they plot observed sediment load against observed flow. The light squares in Figure 11 plot modelled sediment load against modelled flow at exactly the same dates and times at which sediment load measurements were made. The fit between the two datasets is good – at least as good (as is apparent from an inspection of Figure 10) as that between the regressed mathematical representation of the rating curve and the field-measured rating curve.

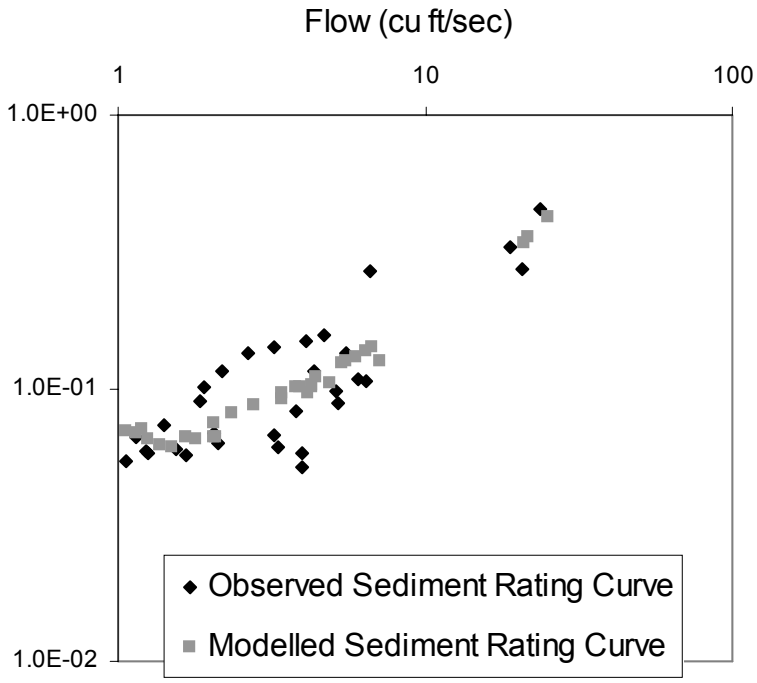


Figure 11. Observed and Modelled Sediment Rating Curves.

If you allowed PEST to run to completion when undertaking the earlier *case4* run, file *case4.par* will contain true values for all parameters, except for *kger* and *jger* which did not feature in that run. An inspection of the *case7* run record file (ie. *case7.rec*) reveals that estimated values are close to these. However it is disturbing to find that for some of these parameters the true parameter value is not within the linear 95% confidence interval calculated as an outcome of the present parameter estimation process.

The true values of *kger* and *jger* are 1.0 and 1.2 respectively. Neither of these fall within the linear 95% confidence intervals of the estimated values of these parameters. An inspection of the run statistics at the end of the PEST control file indicates a moderate degree of correlation between these two parameters (0.84). This will contribute to the uncertainty of their estimation; however this uncertainty is obviously underestimated by the linear-based statistics that are determined as a direct outcome of the parameter estimation process. Hence in order to determine the true uncertainty associated with these parameters and, more importantly, in order to determine the true non-linear uncertainty associated with predictions of sediment load under different land-management or climatic scenarios, true nonlinear predictive uncertainty analysis must be carried out. This can be done using PEST's predictive analyser.