PEST

Surface Water Utilities

Watermark Numerical Computing

and

University of Idaho

August, 2008

PREFACE

The PEST Surface Water Utilities are a series of programs that facilitate the use of PEST with surface water models.

I wish to acknowledge the substantial financial contribution made to the development of the software documented herein by the University of Idaho. In particular, I developed the "flagship" of the utility suite, viz. the time series processor TSPROC while I was working at the Idaho Falls campus of the University of Idaho as a Visiting Research Scientist.

Other organisations that deserve acknowledgment for providing financial assistance toward the development of at least some of the programs documented herein include the United States Environmental Protection Agency (USEPA), the United States Geological Survey (USGS), the Australian Land and Water Resources Research and Development Corporation (LWRRDC), and the Queensland Department of Natural Resources (QDNR). Please report any problems or bugs associated with the use of these utilities to the following email address:-

johndoherty@ozemail.com.au

Dr. John Doherty
Watermark Numerical Computing
Brisbane
Australia

ALPHABETICAL LISTING OF PEST SURFACE WATER UTILITIES

adjobs	Adjusts observation weights for different observation groups in a PEST control file using a simple user-adjustable expression.
iqqm2smp	Converts IQQM output data into site sample file format.
pd_ms1	A PEST driver that undertakes many parameter estimation or predictive analysis runs based on random parameter starting values.
pd_ms2	A PEST driver that combines random sampling and parameter tracking to find the global minimum of the objective function.
pestprp1	Automates construction of PEST instruction and control files for calibration of models which generate output in site sample file format.
plt2smp	Build a site sample file on the basis of a HSPF-generated plot file. Used as part of a composite model run by PEST.
smp2hyd	Rewrites the contents of a site sample file for a user-specified list of sites in a form suitable for plotting against time.
smp2smp	Interpolates data contained within one site sample file to the dates and times represented in another site sample file.
smp2vol	Calculates volumes between arbitrary dates and times for flow samples listed in a site sample file.
smpcal	Calibrates one time series dataset against another.
smpchek	Checks the integrity of a site sample file.
tsproc	A comprehensive time series processor, designed for use as a stand-alone data processor, an environmental model post-processor, and a PEST input file generator.

Table of Contents

Introduction

General

The PEST Surface Water Utilities are a suite of programs whose primary purpose is to assist in the use of PEST with surface water models; however other useful data-processing functions, independent of the calibration process, can also be carried out by many of them. Surface water models are often characterised by the production of lengthy output files containing one or a number of long time series. The data against which the model is to be calibrated is also often voluminous. Because of the amount of data involved, automation of model post-processing tasks and PEST input file preparation is a necessity.

The principal member of the Surface Water Utility suite is TSPROC. In fact, use of TSPROC makes use of some of the other utilities redundant. Nevertheless, all of the original utility programs have been retained within the PEST Surface Water Utility suite, for they may still prove useful in many modelling contexts (especially programs such as SCEUA_P and PD_MS2 which can be used as an alternative to PEST, and as an enhancement to PEST respectively in many calibration contexts). However the importance of TSPROC is reflected in the fact that documentation for this program occupies the bulk of this manual, documentation for the other utility programs being assigned to Appendix A.

Appendix B describes the format of a "site sample file", an ASCII file used by many of the programs described herein for the storage of time series data. The format of this file is such that it can be easily exported from, or imported to, a user's project database, if not directly, then with minimal alterations using a standard text editor.

While most of the programs documented herein are quite general, pertaining to no model in particular, a number are specific to certain models. These model-specific programs transform model output data written in model-specific format to the format required by the utility programs documented herein. Where such a data interface is not provided for a surface water model of particular interest to a specific user, an interface will have to be written by the user him/herself; this is unlikely to be a difficult task. Once this is done, use of PEST with that model can be easily accomplished using the programs documented herein. Alternatively, contact Watermark Numerical Computing; I may be able to write the interface for you.

Installation Instructions

Copy file *swutils.exe* (a self-extracting archive) to a suitable directory on your hard disk (eg. *c:\swutils*). Then run it by tying its name at the command-line prompt. Once the program files have been extracted, you can delete *swutils.exe* to save disk space. Then edit file *autoexec.bat*, adding the name of the utilities' directory to the *path* environment variable and restart your machine.

Source Code and Compilation Details

Source code can be provided for all of the PEST Surface Water Utilities on request. All of the Utilities are written in (almost) ANSI Standard FORTRAN 90; thus, theoretically, they can be compiled to run on any platform for which a FORTRAN 90 compiler is available.

Backtracking

Each of the programs comprising the PEST Surface Water Utilities requires the user to supply information in response to command-line prompts. While this can be a cumbersome method of communication between a program and its user, inconvenience has been mitigated by the fact that a user can always "backtrack" in program execution by entering "e" (for "escape") in response to any prompt. Thus, whether a text string or number is expected following the prompt, a simple "e" will cause the program to display its previous prompt; responding to this prompt with another "e" will make the program display the prompt before that, etc. Hence if, in the process of replying to a succession of screen prompts incorrect data is entered at any stage, a user can "wind the program back" to the point at which previous data entries were all correct, and recommence execution from that point.

Date Format

For programs other than TSPROC (ie. for all of the older programs documented in Appendix A), a particular file named a "settings file", must reside in the directory from they are run. This file must be named *settings.fig*. If a settings file is not present in the working directory, these programs will terminate execution with an appropriate error message.

The presence of a settings file is essential for these older members of the Surface Water Utility suite in order that that they know how to read and write date and time information. Depending on the information contained in the settings file, dates are read and written using either the *dd/mm/yyyy* convention or the *mm/dd/yyyy* convention. The figure below shows a settings file. The format of this file is obvious from this example; it can be written using any text editor.

```
date=mm/dd/yyyy
```

A settings file.

Program TSPROC also needs to know what protocol to use for reading and writing dates. However for TSPROC this information is supplied through its SETTINGS block. Hence the settings file *settings.fig* does not need to be present in the directory from which it is run.

Other File Formats

See Appendix B for the format specifications of a site sample file and a site listing file.

Introduction to TSPROC

General

TSPROC fills two roles. First, it is a time series processor, having the ability to perform many different types of operations on observed and model-generated time series. Second, it automates the generation of PEST input files for calibration tasks of arbitrary complexity based on these time series.

In contrast to the functionality available through other time series analysis software, many of the operations performed by TSPROC are designed specifically for use in the model calibration context. A key element of the processing required in this context is the temporal interpolation of a model-generated time series to the times at which measurements constituting a measurement time series were made. Because measurements of a particular environmental quantity are often intermittent rather than regular, TSPROC does not assume that any individual time series which it manipulates has a constant sample interval. In some instances the absence of this simplifying assumption makes computations carried out by TSPROC a little more inefficient than if time series of constant sample interval were processed. However it does mean that most of the operations carried out by TSPROC are perfectly general in nature.

While TSPROC can be run as an independent executable program, it is also designed to be run as part of a "composite model" by PEST. A "composite model" is a model comprised of two or more executable programs run in succession through a batch or script file. When used in this way TSPROC acts as a model post-processor, carrying out operations of arbitrary complexity on one or many of the time series generated by the model. Similar operations can be carried out on time series comprised of measurement data. The processed measurements and their model-generated counterparts can then be compared, and the discrepancies between the two reduced to a minimum as part of the calibration process undertaken by PEST. In order to facilitate the use of PEST in this context, TSPROC can generate PEST input files appropriate to the type of time series processing that it undertakes as part of the composite model calibrated by PEST.

By using TSPROC it is possible to incorporate some or all of the following data types into the model calibration process.

1. *"Raw data" such as flow and constituent measurements.* Thus field measurements can be compared directly with their model-generated counterparts after the latter have been interpolated to field measurement times.

2. *"Processed data" such as high-pass and low-pass filtered flow time series.* TSPROC includes digital filtering capabilities which allow the separation of high, medium and low frequency components of any time series. This can be useful in baseflow separation; see Nathan and McMahon (1990). Modelled and observed filtered counterparts can be individually matched through the calibration process.

3. *Accumulated volumes and masses.* Using TSPROC, flow volumes and constituent masses can be accumulated between any number of arbitrary dates and times occurring within the model simulation period. It has been found that inclusion of volumetric and mass data, calculated on the basis of field measurements on the one hand and model-generated flows and constituent concentrations (interpolated to field measurement times) on the other hand, can bring numerical stability to the parameter estimation process, and result in more robust estimates of parameter values.

4. *Exceedence-time characteristics.* As with volumetric and mass data, inclusion of exceedence-time characteristics in the inversion process can decrease the likelihood of numerical instability at the same time as it promotes estimation of a realistic set of parameter values. Furthermore, in many modelling applications it is crucial that a model predict exceedence-time characteristics as accurately as possible under future climatic/management conditions. A necessary (though not sufficient) condition for achieving this is that the model replicate these characteristics under historical climatic/management conditions; the latter condition is ensured by including these characteristics in the model calibration process.

5. *Various statistics (mean, sum, maximum, minimum, range and standard deviation) calculated from the terms of a time series (or functions of these terms) over varying time intervals.* Such items as these can be included in the parameter estimation process in their own right (where statistics calculated on the basis of an observed time series are matched with statistics calculated on the basis of the model-generated counterpart to the observed time series), or can be used in conjunction with PEST's predictive analyser. For example, in the latter capacity PEST might be asked to maximise or minimise the maximum value of a possible flow or constituent event, while ensuring that model parameters are such that the model remains in a calibrated state.

6. *Functions of arbitrary complexity calculated on the basis of one or more measured or modelled time series.* In many instances of model calibration it may be better to include a comparison of "derived time series", rather than "raw time series" in the parameter estimation process. To achieve this, TSPROC allows the user to calculate any number of new time series based on relationships of arbitrary complexity between existing time series. For example, in some calibration contexts it may be beneficial to compare the log (or some other function) of a measurement type with its model-generated counterpart over all or part of the model simulation time. Or it may be useful to compare a combination of today's and yesterday's flow with the model-generated equivalent of this same quantity. Minimising the discrepancies between two such "composite time series" may result in better parameter estimates, as well as better estimates of the uncertainties associated with these parameters, because it incorporates the correlation structure of flow and constituent measurements into the parameter estimation process; see, for example, Kuczera (1983).

7. *Data "patterns" and interrelationships pertaining to observed time series and their model-generated counterparts.* Due to the noisy and erratic nature of some types of environmental measurements (particularly those pertaining to some aspects of water quality), it may not be possible to calibrate a model by attempting to directly match field data with their model-generated counterparts. In such situations it may be better to match *relationships* between flow and constituent data calculated on the basis of measurements, with identical relationships calculated on the basis of model outputs. Relationships such as those used by the USGS ESTIMATOR program (Baier et al., 2000) may be suitable in many instances. Implementation of such pattern- or relationship-matching in the parameter estimation process can be accommodated through the use of TSPROC.

Model Calibration using TSPROC

Inclusion of the above (and many other) types of "processed" data in the calibration process is achieved through carrying out the following steps.

1. Process various types of measurement data to generate an appropriate "value-added measurement dataset".

2. Time-interpolate "raw" model-generated time series to the times at which field measurements were made; then process that data in an identical fashion to that in which the measurement data were processed.

3. Generate a PEST input dataset; this is comprised of a PEST control file recording the value-added measurements used in the calibration process, and an instruction file capable of reading the model-generated counterparts to these "measured" quantities from the appropriate model output file.

The first of the above steps is easily carried out using TSPROC. As presently coded, TSPROC can read field measurements from either a WDM file or from a "site sample file" (see Appendix B). The second of the above operations can be carried out just as easily, provided model-generated time series are recorded in either of these two file formats. Note, however, that TSPROC can also read model-generated time series from a HSPF PLOTGEN file.

As mentioned above, when a model is being calibrated by PEST, TSPROC should be run following the model as part of a batch or script file run by PEST as a "composite model". Hence the "model output file" in this case will, in fact, be a TSPROC output file. This file will contain the model-generated equivalents of the processed field measurements produced through the first of the above steps. The role of the calibration process is then to minimise the discrepancies between these two data sets.

TSPROC can also be used to carry out the third of the above tasks. If provided with the set of template files pertinent to the current calibration exercise (these carrying the names of the parameters to be adjusted through the parameter estimation process), TSPROC will write a complete PEST control file in which these parameters are recorded, together with the (processed) measurements to which model-generated equivalents must be matched through

the parameter estimation process. In doing this, TSPROC will assign names to all observations involved in the parameter estimation process, and assign weights to these observations according to formulas of arbitrary complexity supplied by the user; a different formula can be supplied for each measurement type. TSPROC will then write the instruction file by which PEST can read the model-generated equivalents to these (processed) measurements from a TSPROC output file when the latter is run as part of a composite model by PEST.

As can be seen from this brief description, TSPROC is a program of considerable complexity. It was written in order to provide a tool by which most of the arduous data-handling tasks required to calibrate a surface water model can be carried out automatically, thus making PEST setup relatively easy in this context. Furthermore, its design is such as to allow a large degree of flexibility in the way this process is carried out, thus allowing a modeller to tailor the parameter estimation process to the demands of his/her particular modelling application. As time goes on, and as more experience is gained in applying PEST to surface water model calibration, the functionality included in TSPROC will be increased accordingly. This is only the beginning.

Because of the multiplicity of tasks which it undertakes, TSPROC can be used in place of many of the older programs of the PEST Surface Water Utility suite. Nevertheless, as mentioned above, these older programs are retained in the suite because of their inherent usefulness in performing certain specific tasks required in surface water model calibration.

PAR2PAR

It is worth noting that TSPROC is a useful complement to the utility program PAR2PAR supplied with PEST. The latter program performs arbitrary manipulation of model *parameters* (in contrast to TSPROC, which manipulates model *outputs*). Using PAR2PAR, native model parameters can be calculated from parameters that PEST "sees" (ie. the parameters that are actually adjusted as part of the parameter estimation process) using mathematical equations involving one or many parameters. By using PAR2PAR, the following strategies can be implemented in the parameter estimation process:-

1. *Estimation of "super parameters" from which "native model parameters" are calculated.* For example, if monthly variation of a parameter can be characterized by the mean value, amplitude and phase of a sine wave, the number of parameters required to characterize seasonal variation of a particular process can be reduced from 12 to 3. This (and similar) reductions in the numbers of parameters can make the calibration process much more tractable than it otherwise would be.

2. *Relationships between soil properties and native model parameters can be incorporated into the inversion process.* The regression coefficients featured in such relationships can be estimated together with, or instead of, native model parameters through the calibration process. In some instances this can result in a substantial reduction in the numbers of parameters requiring estimation, at the same time as it ensures that parameters are assigned physically and hydraulically realistic values.

3. *Realistic ordering relationships between parameters can be enforced by estimating parameter ratios, rather than native model parameters.* Through this mechanism, in combination with PEST's unique parameter bounding functionality, a lower bound on one parameter can be set at the current value of another parameter. Thus the corresponding native model parameters will always take values which obey the correct ordering relationship.

PEST-ASP

The ultimate "weapon" in the surface water model calibration "arsenal" available through PEST and its Surface Water Utilities is PEST itself, or rather the latest version of PEST, named PEST-ASP. Functionality available through PEST-ASP includes a powerful mode of operation known as "regularisation mode" – see the PEST manual for further details. Use of PEST in regularisation mode allows the estimation of many more parameters than is possible using traditional nonlinear parameter estimation techniques and software, a capability that serves PEST well when used in surface water model calibration, as many of the models used in this field are very highly parameterised. PEST is able to work in such highly-parameterised contexts by calculating parameter values which deviate from a user-defined "default condition" only to the minimum extent necessary to achieve a good fit between model outputs and field data. The "default condition" can be supplied in the form of preferred values of parameters, or of relationships between parameters. Thus the deleterious effects of parameter nonuniqueness that always attend the simultaneous estimation of too many parameters (viz. numerical instability and/or the estimation of wild and erratic parameter values), are obviated as parameter uniqueness is ensured by reference to the "default condition".

Another item of PEST-ASP functionality that is also particularly useful in the surface water modelling context is PEST's predictive analyser. When used in "predictive analysis mode", PEST is able to maximise or minimise a key model prediction under the constraint that the model remain in a calibrated state. In some surface water modelling applications, particularly those associated with water quality, this can be very useful indeed, for in situations such as these the complexity of the simulated processes requires that many parameters be introduced to the model. Rarely, however, does the calibration dataset allow unique estimation of these parameters. The resulting parameter nonuniqueness may then result in a large degree of uncertainty associated with key water quality predictions. The magnitude of this uncertainty can be fully explored using PEST's predictive analyser.

Using TSPROC

Running TSPROC

TSPROC is run by typing its name at the screen prompt.

TSPROC prompts for only two items of information. The first is the name of its input file; the second is the name of its run record file. The TSPROC input file contains all of the information required for TSPROC to perform the various operations for which it was designed. As it carries out these operations it echoes the contents of its input file, and the operations that it performs in response to the instructions provided in that file, to the screen and to its run record file. A TSPROC input file is easily prepared using a text editor.

If TSPROC is requested to generate a set of PEST input files it will prompt the user before overwriting any existing files of the same name. For example, it may prompt:-

```
File instruct.ins already exists.  Overwrite it?  [y/n]:
```

Type "y" or "n", followed by <Enter> as appropriate. Note that TSPROC does not prompt in this manner when overwriting data files as part of its time series manipulation functionality. This is because, if it is run many times in the course of a PEST run, such files will need to be overwritten on each occasion that it is run. However if the user forgets to de-activate instructions within a TSPROC input file for generation of PEST input files (see below) before TSPROC is used by PEST in a parameter estimation run, then the above warning message may save previously-generated PEST input files (perhaps those being used for the current PEST run) from being overwritten.

If TSPROC is run by PEST as part of a composite model, then the responses to TSPROC's prompts must be placed into a text file prior to the PEST run and provided to TSPROC through the command-line re-direction mechanism. For example, if it is desired that TSPROC read an input file named *tsproc.dat* and that it record details of its operations to a run record file named *tsproc.rec*, then a text file (named, for example, *tsproc.in*) should be prepared with its contents as follows:-

```
tsproc.dat
tsproc.rec
```

Contents of a text file containing the responses to TSPROC prompts.

When TSPROC is then run as part of a composite model by PEST, the pertinent line in the composite model batch file should be:-

```
tsproc < tsproc.in
```

(The "<" symbol instructs TSPROC to look for its keyboard input from the ASCII file whose name follows it.) Of course the above command can be issued from the keyboard as well if file *tsproc.in* has already been prepared.

Because TSPROC is a complex program which carries out many different types of operations, its input file must be carefully and thoughtfully prepared. It is not impossible that a user will make an error in preparation of this file. Should this occur, TSPROC will report the error to the screen, and to its run record file, and then cease execution; it will not read its input file any further, nor perform any operations beyond that at which the error occurred. Thus while TSPROC's error-checking functionality is quite comprehensive, it will only find one error at a time in a TSPROC input file that contains multiple errors.

In some circumstances a user may desire that TSPROC not report its activities to the screen, for example if TSPROC is being run under the control of PEST and the user wishes that TSPROC screen output not interfere with that of PEST. As with any command-line program, TSPROC output can be re-directed from the screen to a file, thus leaving the screen bare. Because the TSPROC run record file contains all of the information that TSPROC writes to the screen, there is nothing to be gained through keeping such a file which contains re-directed screen output; hence it is best to re-direct TSPROC screen output to the "nul" file, (ie. to nowhere). Hence if it is desired that TSPROC look to a file named *tsproc.in* for its keyboard input, and that it re-direct its screen output to the "nul" file, it should be run using the command:-

```
tsproc < tsproc.in > nul
```

The TSPROC Input File - Overview

The TSPROC input file is divided into a series of sections or "blocks". Within each block, various items of information are supplied following pertinent "keywords" which identify each such item. In most blocks these keywords can be supplied in any order; however there are some exceptions to this rule which will be pointed out in the pertinent sections of this manual. (TSPROC will also inform you, through an appropriate error message, if keyword ordering is incorrect.) Keywords are shown capitalised in the illustrations used throughout this document for ease of recognition. Note, however, that the contents of a TSPROC input file are case-insensitive.

Any line within a TSPROC input file beginning with the "#" character is ignored. Thus comments can be freely interspersed with data elements in a TSPROC input file. The complex nature of the instructions that can be supplied to TSPROC through its input file makes the inclusion of comments in this file a good idea.

An example of a TSPROC input file follows.

```
START SETTINGS
  CONTEXT pest_input
  DATE_FORMAT mm/dd/yyyy
END SETTINGS


#####################################################################
# Modelled river flows are read from a HSPF output file.
#####################################################################

START GET_SERIES_PLOTGEN
```

```
  CONTEXT all
  FILE catchment.plt
  LABEL "total outflow"
  NEW_SERIES_NAME flow_mod
END GET_SERIES_PLOTGEN


#########################################################################
# Observed river flows are read from a WDM file.
#########################################################################

START GET_SERIES_WDM
  CONTEXT all
  FILE catchment.wdm
  DSN 113
  NEW_SERIES_NAME flow_obs
END GET_SERIES_WDM


#########################################################################
# Modelled flows are interpolated to the times of observed flows.
#########################################################################

START NEW_TIME_BASE
  CONTEXT all
  SERIES_NAME flow_mod
  TB_SERIES_NAME flow_obs
  NEW_SERIES_NAME i_flow_mod
END NEW_TIME_BASE


#########################################################################
# Flow volumes are accumulated for the modelled time series.
#########################################################################

START VOLUME_CALCULATION
  CONTEXT all
  SERIES_NAME i_flow_mod
  NEW_V_TABLE_NAME vol_mod
  FLOW_TIME_UNITS days
  DATE_FILE dates.dat
END VOLUME_CALCULATION

#########################################################################
# Flow volumes are accumulated for the observed time series.
#########################################################################

START VOLUME_CALCULATION
  CONTEXT pest_input
  SERIES_NAME flow_obs
  NEW_V_TABLE_NAME vol_obs
  FLOW_TIME_UNITS days
  DATE_FILE dates.dat
END VOLUME_CALCULATION


#########################################################################
# Exceedence times are calculated for the modelled time series.
#########################################################################

START EXCEEDENCE_TIME
  CONTEXT all
```

```
  SERIES_NAME i_flow_mod
  NEW_E_TABLE_NAME time_mod
  EXCEEDENCE_TIME_UNITS days
  FLOW 0
  FLOW 10
  FLOW 20
  FLOW 50
  FLOW 100
  FLOW 200
END EXCEEDENCE_TIME

#########################################################################
# Exceedence times are calculated for the observed time series
#########################################################################

 START EXCEEDENCE_TIME
  CONTEXT pest_input
  SERIES_NAME flow_obs
  NEW_E_TABLE_NAME time_obs
  EXCEEDENCE_TIME_UNITS days
  FLOW 0
  FLOW 10
  FLOW 20
  FLOW 50
  FLOW 100
  FLOW 200
END EXCEEDENCE_TIME


#########################################################################
# Modelled time series and tables are written to a file.
#########################################################################

START LIST_OUTPUT
  CONTEXT all
  FILE model.out
  SERIES_NAME i_flow_mod
  V_TABLE_NAME vol_mod
  E_TABLE_NAME time_mod
  SERIES_FORMAT short
END LIST_OUTPUT_BLOCK


#########################################################################
# PEST input files are written.
#########################################################################

START WRITE_PEST_FILES
  CONTEXT pest_input
  NEW_PEST_CONTROL_FILE case.pst
  TEMPLATE_FILE catchment.tpl
  MODEL_INPUT_FILE catchment.uci
  TEMPLATE_FILE extra.tpl
  MODEL_INPUT_FILE extra.dat
  NEW_INSTRUCTION_FILE observation.ins
  AUTOMATIC_USER_INTERVENTION yes

########### Time series observations  ######

  OBSERVATION_SERIES_NAME flow_obs
  MODEL_SERIES_NAME i_flow_mod
  SERIES_WEIGHTS_EQUATION 1.0/sqrt(@_abs_value)
```

```
  SERIES_WEIGHTS_MIN_MAX 1.0 100.0

############# volumes ######################

  OBSERVATION_V_TABLE_NAME vol_obs
  MODEL_V_TABLE_NAME vol_mod
  V_TABLE_WEIGHTS_EQUATION 5.0

############# exceedence-times #############

  OBSERVATION_E_TABLE_NAME time_obs
  MODEL_E_TABLE_NAME time_mod
  E_TABLE_WEIGHTS_EQUATION log(2.0/@_abs_value) + 2.0
  E_TABLE_WEIGHTS_MIN_MAX 0 1000

############ other data ####################

  PARAMETER_DATA_FILE param.dat

END WRITE_PEST_FILES
```

A TSPROC Input File

Each block within a TSPROC input file instructs TSPROC to carry out a certain type of operation. Information supplied within a block informs TSPROC of the names of the entities to be processed, and the names of the entities to be produced as a result of that processing. Any other information required by TSPROC to enable that processing to take place is also supplied within the block through the appropriate keyword. For each block some keywords are optional and some are mandatory. Where an optional keyword is not supplied TSPROC supplies a default value for its associated variable.

With one exception (see below) blocks can be arranged in a TSPROC input file in any order. However because TSPROC processes blocks in the order in which they are supplied, the ordering of blocks can be important in many applications (for example if an entity that is produced in one block is used by another block).

The DATE_FORMAT and CONTEXT Settings

In any TSPROC input file, there is one block which must be present, and which must precede all other blocks. This is the SETTINGS block. The SETTINGS block must contain two keywords, viz. the DATE_FORMAT and CONTEXT keywords.

The DATE_FORMAT keyword informs TSPROC of the protocol to be used for representation of dates in all input files which it reads and output files which it generates. Only two options are presently available viz. *dd/mm/yyyy* and *mm/dd/yyyy*.

The CONTEXT keyword must be followed by a character string of 20 characters or less (with no embedded spaces) which “sets the context” for the current TSPROC run. A CONTEXT keyword is also a mandatory element of every other block appearing in a TSPROC input file; as in the SETTINGS block, the CONTEXT keyword in all of these blocks must be followed by a string of 20 characters or less. Up to five CONTEXT keywords can appear in any TSPROC processing block. (A “processing block” is any block other than

the SETTINGS block.) If the CONTEXT string following any of the CONTEXT keywords in a processing block agrees with that in the SETTINGS block, then the instructions in that block will be implemented by TSPROC. If not, they will be ignored (unless at least one of the CONTEXT strings supplied in a processing block is "all", in which case the operations listed in the block will be carried out regardless of the current TSPROC context as defined in the SETTINGS block). Furthermore CONTEXT keywords must precede all other keywords in the block. Use of the CONTEXT concept allows a user to "turn on" and "turn off" various processes cited in a TSPROC input file, simply by altering the CONTEXT string in the SETTINGS block. This can be very useful when preparing for a PEST run.

Blocks within a TSPROC Input File

The following table lists the blocks which may be present within any TSPROC input file. Multiple occurrences of any block except the SETTINGS block are permitted.

Block name	**Function of Block**
DIGITAL_FILTER	Passes a time series through a high-pass, low-pass or band-pass digital butterworth filter, or a "base flow separation filter", to produce a new time series.
ERASE_ENTITY	Removes a time series, c_table, s_table, v_table or e_table from TSPROC memory.
EXCEEDENCE_TIME	Calculates the times over which terms of a time series exceed user-specified thresholds, thus creating an e_table.
GET_SERIES_PLOTGEN	Imports one or a number of time series from a HSPF PLOTGEN file.
GET_MUL_SERIES_GSFLOW_GAGE	Imports one or a number of series from "gage files" produced by the USGS GSFLOW model.
GET_MUL_SERIES_SSF	Imports multiple time series from a site sample file.
GET_MUL_SERIES_STATVAR	Imports one or a number of series from "statvar files" written by the PRMS model.
GET_SERIES_SSF	Imports a time series from a site sample file.
GET_SERIES_TETRAD	Imports one or a number of time series from a TETRAD output file.
GET_SERIES_UFORE_HYDRO	Imports a time series from a UFORE_HYDRO model input or output file.
GET_SERIES_WDM	Imports a time series from a WDM file.
LIST_OUTPUT	Writes TSPROC time series, c_tables, s_tables, v_tables and e_tables to a text file.
NEW_SERIES_UNIFORM	Creates a uniform valued series with a constant time increment.

NEW_TIME_BASE	Interpolates one time series to the sample dates and times of another.
REDUCE_TIME_SPAN	Shortens a time series by deleting terms outside a user-specified date/time interval.
SERIES_BASE_LEVEL	Subtracts a single term of one time series from all terms of another time series.
SERIES_CLEAN	Erases terms in a series between user-supplied thresholds.
SERIES_COMPARE	Calculates statistics which describe the goodness of fit between one time series and another, placing the results in a c_table.
SERIES_DIFFERENCE	Computes a new time series as the difference of subsequent terms of an existing time series.
SERIES_DISPLACE	Advances or retards the terms of a time series by a multiple of the sample interval.
SERIES_EQUATION	Carries out mathematical operations of arbitrary complexity between the terms of any number of time series of identical time base to create a new time series.
SERIES_STATISTICS	Calculates certain statistics based on some or all of the terms comprising a time series, thus creating an s_table.
SETTINGS	Provides settings for the current TSPROC run.
V_TABLE_TO_SERIES	Copies data from a v_table to a time series.
VOLUME_CALCULATION	Calculates volume or mass by time-integration of a flow or flux time series, thus creating a v_table.
WRITE_PEST_FILES	Generates a PEST control file and a PEST instruction file for a parameter estimation process which includes any number of time series, s_tables, v_tables and e_tables.

Blocks occurring within a TSPROC input file.

Each of the blocks present within a TSPROC input file is discussed in detail in the following sections.

TSPROC Entities

Most of the tasks carried out by TSPROC are related to the processing of time series. As stated above, these time series may, or may not, be of constant sample interval. A time series can be comprised of as little as one sample, or as many as tens of thousands of samples.

Each time series must be given a name by the user when it is imported into TSPROC or produced as an outcome of the processing encapsulated in a TSPROC processing block; a

time series is normally named using a NEW_SERIES_NAME keyword. A time series name must be 10 characters or less in length. (At first sight it might appear that 10 characters is unduly restrictive for the name of a time series. The reason for this restriction in length is based on the fact that observation and observation group names used in a TSPROC-generated PEST control file must be formed from the names of these TSPROC entities. For lengthy time series, observations can number in the tens of thousands; TSPROC creates observation names by appending the time series term number to the time series name, or a contraction of the time series name if appropriate. The chances of observation name nonuniqueness are considerably reduced if time series names are restricted to 10 characters in length within TSPROC, thus requiring that the user maintain uniqueness in nomenclature at the 10 character level. Nevertheless, should duplicate observation names be created as a result of its name formation process, TSPROC will detect this and generate an appropriate error message. If the shortness of a time series name prevents an adequate characterisation of the source of each time series, the user is advised to tabulate the hereditary of each time series on a piece of paper.)

Many of the processing options provided by TSPROC produce a new time series through the processing or manipulation of one or a number of existing time series. Where this occurs the user must provide the name of both the existing time series (through a SERIES_NAME keyword) and the new time series (through a NEW_SERIES_NAME keyword) to the processing block through which the operation is being undertaken.

Sometimes the processing of a time series results in the creation of an entity which is not another time series. When TSPROC calculates certain statistics pertaining to the terms of a time series (through the SERIES_STATISTICS block), these statistics are stored in an "s_table". The outcomes of volumetric calculations carried out by the VOLUME_CALCULATION block are stored in a v_table. The outcomes of exceedence time calculations carried out by the EXCEEDENCE-TIME block are stored in an e_table. Statistics based on the comparison of two time series are written to a "c_table". Like the time series entity, each of these other entities must be assigned a name of 10 characters or less in length, this name being provided by the user following the NEW_C_TABLE, NEW_S_TABLE, NEW_V_TABLE and NEW_E_TABLE keywords in the pertinent processing blocks. More entities will probably be added to TSPROC over time as the need arises.

TSPROC will never overwrite one entity with another. Hence the name provided for a new entity in a processing block must be different from the name of any existing entity of the same type. If desired, entities can be erased from memory in order to make room for other entities using the ERASE_ENTITY block. This functionality can be very important when processing lengthy time series which make large demands on computer memory.

Each TSPROC block is now discussed. Descriptions are arranged in alphabetical order.

DIGITAL_FILTER

The DIGITAL_FILTER block instructs TSPROC to calculate a new time series from an existing time series by passing the latter through a digital filter. Two types of filter are provided. The Butterworth filter can remove high frequency components (low pass filter), low frequency components (high pass filter), or both of these (band pass filter) from the original time series. The "baseflow separation" filter allows extraction of quick response from a flow time series; baseflow can then be obtained by subtraction from the original series using the SERIES_EQUATION block.

The nature of digital filtering is such that it can only be performed on a time series for which the sample interval is constant. Thus before performing filtering operations TSPROC checks the nominated time series for this condition; if it is not met, TSPROC terminates execution with an error message. (Use the NEW_TIME_BASE block in conjunction with the NEW_SERIES_UNIFORM block to create a time series interpolated to a uniform time base if this is a problem.)

Keywords available in the DIGITAL_FILTER block are listed in the following table. Two examples of a DIGITAL_FILTER block follow that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the DIGITAL_FILTER block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series on which filtering operations will be carried out.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_SERIES_NAME	*Mandatory.* The name of the new series created by TSPROC through filtering of an existing time series.	Any character string without spaces up to 10 characters in length.
FILTER_TYPE	*Mandatory*. The type of filter being implemented.	"butterworth" or "baseflow_separation"
FILTER_PASS	*Mandatory if FILTER_TYPE is "butterworth"; disallowed otherwise*. Informs TSPROC whether to carry out low, band or high pass filtering.	"low", "band" or "high.

CUTOFF_FREQUENCY	*Mandatory if FILTER_TYPE is "butterworth" and FILTER_PASS is "high" or "low"; disallowed otherwise*. For a high pass filter the 3db point of low frequency rolloff. For a low pass filter the 3db point of high frequency rolloff. Frequency in $days^{-1}$.	Real number.
CUTOFF_FREQUENCY_1	*Mandatory if FILTER_TYPE is "butterworth" and FILTER_PASS is "band"; disallowed otherwise*. The 3db point of low frequency rolloff. Frequency in $days^{-1}$.	Real number.
CUTOFF_FREQUENCY_2	*Mandatory if FILTER_TYPE is "butterworth" and FILTER_PASS is "band"; disallowed otherwise*. The 3db point of high frequency rolloff. Frequency in $days^{-1}$.	Real number.
STAGES	*Optional if FILTER_TYPE is "butterworth"; disallowed otherwise.* Number of filter stages. The more stages, the steeper is the high and/or low frequency rolloff. Default is 1.	Integer. 1, 2 or 3.
ALPHA	*Mandatory if FILTER_TYPE is "baseflow_separation"; disallowed otherwise.* The assumed relative decay rate of baseflow.	A real number greater than zero (normally in the range 0.9 to 0.975).
PASSES	*Optional if FILTER_TYPE is "baseflow_separation"; disallowed otherwise.* The number of filter passes. Default is 1.	Integer. 1 or 3 only.
REVERSE_SECOND_STAGE	*Optional*. If FILTER_TYPE is set to "butterworth", STAGES is set to 2, and FILTER_PASS is set to "low", then the second filter pass is performed in the reverse direction, thereby nullifying any phase shift incurred in the first low pass filter pass.	"yes" or "no"
CLIP_INPUT	*Optional for baseflow separation filter type; disallowed for butterworth.* If activated, prevents terms of filtered time series from exceeding terms of original time series. Default is "no".	"yes" or "no"

CLIP_ZERO	*Optional for baseflow separation filter type; disallowed for butterworth.* If activated, prevents terms of filtered time series from becoming negative. Default is "no".	"yes" or "no"

Keywords in a DIGITAL_FILTER block.

```
START DIGITAL_FILTER
  CONTEXT context_1
  FILTER_TYPE butterworth
  SERIES_NAME flow
  NEW_SERIES_NAME av_flow
  FILTER_PASS low
  CUTOFF_FREQUENCY 0.08
END DIGITAL_FILTER
```

A DIGITAL_FILTER block in which a Butterworth filter is implemented.

```
START DIGITAL_FILTER
  CONTEXT context_1
  FILTER_TYPE baseflow_separation
  SERIES_NAME flow
  NEW_SERIES_NAME qflow
  ALPHA 0.95
  PASSES 1
  CLIP_INPUT yes
  CLIP_ZERO yes
END DIGITAL_FILTER
```

A DIGITAL_FILTER block in which a baseflow separation filter is implemented.

Digital filtering is a fast and powerful means of accentuating certain aspects of a time series and removing others. A high pass filter removes long-term variations from a time series, while a low pass filter removes short term variations. A band pass filter removes both short and long-term variations, allowing only medium-term variations to remain in the filtered time series. Many different types of filters can be constructed to implement all three of these types of operation. TSPROC implements the Butterworth filter; this has the desirable property that its frequency response is maximally flat within the pass band. It also implements a "baseflow separation" filter – a form of high pass filter with a more gentle frequency rolloff than the Butterworth filter outside the pass band. This is suitable for separation of the quickflow component of streamflow; baseflow can then be obtained by subtraction from the original streamflow. See Nathan and McMahon (1990) for details.

Frequencies within the pass band of a filter are conveyed with minimal attenuation. However as the edge of the pass band is approached, and outside the passband, attenuation of the input

time series takes place. The diminution of output amplitude with increasing or decreasing frequency outside the passband is referred to as "rolloff" in filtering jargon. The more stages that a filter employs, the steeper is this rolloff. However steep rolloff comes at a price – this being the tendency for the filter output to oscillate or "ring" in response to high amplitude events within the input time series. A phase delay between the input and output time series can also be introduced. For a 1-stage Butterworth filter the rolloff is 6db/octave; for a 2-stage Butterworth filter it is 12 db/octave, while for a 3-stage Butterworth filter it is 18 db/octave. Rolloff is 3db/octave for one pass of the baseflow separation filter, and 9db per octave for 3 passes of this filter. An octave is a doubling of frequency; a db is a measure of signal power gain or loss. A rolloff rate of 6db/octave is equivalent to a halving of output amplitude with every factor of two change in frequency. (This is sufficient, or more than sufficient, for most applications in surface water hydrology.)

Use of each of the types of digital filter implemented by TSPROC is now discussed in detail.

Butterworth Filter

When using a Butterworth filter, the frequency characteristics of the filter must be provided directly through pertinent keywords within the DIGITAL_FILTER block.

The boundary between the passband and the stopband of a filter is normally denoted by the "3db point". This is the frequency at which the amplitude response is a factor of about √2 less than it is in the pass band. In designing a low pass Butterworth filter, one such frequency is required; this is supplied with the CUTOFF_FREQUENCY keyword. The same holds for a high pass Butterworth filter, except that the amplitude rolls off with decreasing frequency from the 3db point for a high pass filter whereas it rolls off with increasing frequency from the 3db point for a low pass filter. For a band pass filter an upper and lower 3db frequency are required. These must be supplied following the CUTOFF_FREQUENCY_1 and CUTOFF_FREQUENCY_2 keywords. The former must be less than the latter or TSPROC will terminate processing of the DIGITAL_FILTER block with an appropriate error message.

Frequencies must be supplied in units of day^{-1} no matter what the time increment of the time series. Sometimes it is easier to think in terms of period rather than frequency; period is the reciprocal of frequency. A fluctuation which repeats itself every *n* days has a frequency of *1/n* day^{-1}. *n* can be greater or less than a day. For a period of 6 hours *n* is ¼ days and the frequency is 4 day^{-1}; for a period of 10 days, the frequency is 1/10 day^{-1}.

A high, low or band pass cutoff frequency must be less than one half the sample frequency of the time series which is undergoing filtering. Thus, for example, a cutoff frequency for an hourly time series must be less than 12 day^{-1}. A cutoff frequency for a daily time series must be less than 0.5 day^{-1}.

As mentioned above, steeper frequency rolloff can be achieved through using more than one filter STAGE; up to three STAGEs are allowed by TSPROC. However if a STAGE keyword is not supplied, a single stage is assumed. While more stages mean greater signal rejection within the frequency stopband, the resulting propensity for "ringing", and the greater phase lag between the input and output signals, may be unwanted in many hydrologic applications.

Baseflow Separation Filter

Only two keywords are required to specify the characteristics of a baseflow separation filter. These are the ALPHA and PASSES keywords. ALPHA is the rate of decay of baseflow relative to current flow rate; a value of 0.92 to 0.98 is suitable for most applications; however as pointed out by Nathan and McMahon (1990), a little trial and error may be required for selection of the most appropriate value for any particular application. Its value is independent of the series sample interval. PASSES is similar to the STAGE keyword required by the Butterworth filter. However it is also a little different in that, unlike the Butterworth filter, different internal filter coefficients are not used for different passes. Furthermore, only 1 or 3 passes can be implemented, with the second pass being implemented in the reverse direction to mitigate phase shifts. If the PASSES keyword is not supplied, a value of 1 is assumed.

The outcome of implementation of a baseflow separation filter is a time series which represents the "quick response" streamflow. Baseflow can then be obtained by subtracting this from the original streamflow time series using the SERIES_EQUATION block. Occurrence of subzero filtered terms, or terms which are greater than the original streamflow record, can be prevented by clipping – see below.

Clipping

The outputs of the baseflow separation filter (but not the Butterworth filter) can be clipped in order to prevent the occurrence of negative values, or of values which are greater than those of the input time series. Sub-zero values can be prevented using the CLIP_ZERO keyword, and values which are higher than the input time series can be prevented using the CLIP_INPUT keyword; in either case a "yes" or "no" specifier must be provided in the DIGITAL_FILTER block. A default of "no" is assumed in either case. Clipping is often very useful in conjunction with baseflow separation filtering. It should be remembered, however, that the action of this filter is to provide time series which have similar characteristics to baseflow and quickflow. This, indeed, can be extremely helpful in calibration of a model where the contribution of both of these to the objective function can be monitored (and enhanced if desired through appropriate weights selection). However it should not be forgotten that the calibrated model is then likely to produce a better quickflow/ baseflow time series than the digital filter used to assist in the calibration process.

Settling Time

You should be aware of the fact that a filter sometimes takes a while to "settle down" when filtering operations begin on a time series. This will apply more to a multi-stage Butterworth filter than to the other filter types implemented by TSPROC. To ensure integrity of a filtered time-series it may sometimes be necessary to remove the first part of the series using the REDUCE_TIME_SPAN block. Note also, that TSPROC will not allow filtering operations to take place on any time series that has fewer than 20 entries.

As mentioned above, filtering can only be implemented on a time series in which the sample interval is constant throughout the series. TSPROC will report any attempt to filter a time series with non-constant sample interval.

Reverse Filtering

If FILTER_TYPE is set to “butterworth”, FILTER_PASS is set to “low” and STAGES is set to 2, the second stage of Butterworth filtering can be performed in the reverse direction to that of the first stage of filtering. Low pass filtering can incur a substantial phase shift, thereby delaying peaks and troughs occurring within the original time series. This phase-change-induced delay can be rectified by running the digital filter from late times to early times in the second filtering stage.

The user should use this option with caution. It can sometimes actually amplify the low-frequency component of filtered peaks and troughs remaining after the two-stage filtering operation has taken place.

ERASE_ENTITY

If a time series, c_table, s_table, v_table or e_table is no longer required by TSPROC, it can be erased from TSPROC's memory in order to make room for other TSPROC entities. This may be a wise thing to do if a time series which contains many terms is no longer required. This is achieved through use of the ERASE_ENTITY block. Keywords found in the ERASE_ENTITY block are listed in the table below; an example of an ERASE_ENTITY block follows that.

Keywords in the ERASE_ENTITY block can be supplied in any order except for the CONTEXT keyword(s), which must precede all other keywords.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the ERASE_ENTITY block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Optional.* The name of a time series to be erased.	A name of 10 characters or less in length referencing an time series stored within TSPROC's memory.
C_TABLE_NAME	*Optional*. The name of a c_table to be erased.	A name of 10 characters or less in length referencing a c_table stored within TSPROC's memory.
S_TABLE_NAME	*Optional*. The name of an s_table to be erased.	A name of 10 characters or less in length referencing an s_table stored within TSPROC's memory.
V_TABLE_NAME	*Optional*. The name of a v_table to be erased.	A name of 10 characters or less in length referencing a v_table stored within TSPROC's memory.
E_TABLE_NAME	*Optional.* The name of an e_table to be erased.	A name of 10 characters or less in length referencing an e_table stored within TSPROC's memory.

Keywords in an ERASE_ENTITY Block.

```
START ERASE_ENTITY
  CONTEXT context_1
  C_TABLE_NAME compare
  E_TABLE_NAME ex_flow
  S_TABLE_NAME stat_flow
  V_TABLE_NAME vol_flow
  SERIES_NAME flow
END ERASE_ENTITY
```

An ERASE_ENTITY block.

EXCEEDENCE_TIME

The EXCEEDENCE_TIME block instructs TSPROC to calculate the time over which user-supplied flows or fluxes have been exceeded, or over which such nominated flows or fluxes have not been exceeded. The outcomes of EXCEEDENCE_TIME calculations are stored in an e_table. Like every other storage entity used by TSPROC, the user must provide a name for each e_table produced in this manner so that it can be referenced in later processing.

Keywords available in the EXCEEDENCE_TIME block are listed in the following table; an example of an EXCEEDENCE_TIME block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others, and the DELAY keyword which (if used) must directly follow a FLOW keyword.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the EXCEEDENCE_TIME block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series on which exceedence time calculations will be carried out.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_E_TABLE_NAME	*Mandatory.* The name of the new e_table used to store the outcomes of exceedence-time calculations carried out by TSPROC.	Any character string without spaces up to 10 characters in length.
EXCEEDENCE_TIME_UNITS	*Mandatory*. The time units to be used for storage of calculated exceedence times.	"year", "month", "day", "hour", "min" or "sec"
UNDER_OVER	*Optional.* Informs TSPROC whether to calculate times for which flow thresholds are exceeded ("over") or are not exceeded ("under").	"under" or "over"; default is "over".
FLOW	*At least one FLOW keyword must be present*. This is a threshold for which exceedence times are to be calculated.	Real number.

DELAY	*Optional; but if supplied for one FLOW, must be supplied for all FLOWs.* The time delay for any one event before time accumulation commences.	Real number. Time units are same as those supplied with EXCEEDENCE_TIME_UNITS keyword.

Keywords in an EXCEEDENCE_TIME block.

```
START EXCEEDENCE_TIME
  CONTEXT all
  SERIES_NAME outflow
  NEW_E_TABLE_NAME et_flow
  EXCEEDENCE_TIME_UNITS days
  FLOW 0.0
  FLOW 10.0
  FLOW 20.0
  FLOW 50.0
  FLOW 100.0
  FLOW 200.0
END EXCEEDENCE_TIME
```

An EXCEEDENCE_TIME block.

Any number of FLOW keywords can be provided in an EXCEEDENCE_TIME block. For each such FLOW, if UNDER_OVER is set to "over" (or if this keyword is omitted) and if no DELAY keywords are supplied, TSPROC calculates the accumulated time over which the nominated flow was exceeded. Alternatively, if UNDER_OVER is set to "under", TSPROC calculates the accumulated time for which flow was less than each nominated FLOW. Note that in carrying out these calculations TSPROC does more than simply count the number of time series terms which exceed, or are less than, the value of each FLOW, and then multiply the number of terms by the series sampling interval. This would be an incorrect procedure for two reasons. The first of these reasons is that, as mentioned above, TSPROC does not assume a uniform sampling interval for any series. The second reason is that an exceedence-time calculation that is carried out in this way on the basis of a model-generated time-series will be slightly discontinuous with respect to model parameters (which will lead to a degradation in the performance of PEST as it attempts to estimate these parameters). Instead, TSPROC carries out linear interpolation between the terms of a time series to find the "exact time" at which a FLOW threshold was crossed, and commences or ceases time-accumulation from that point. The result is a continuous relationship between exceedence times and parameters as the latter vary during a parameter estimation process.

Exceedence times calculated by TSPROC can be stored internally (and listed through the LIST_OUTPUT block) in time units of years, months, days, hours, minutes or seconds. The user must choose one of these options through the mandatory EXCEEDENCE_TIME_UNIT keyword.

Note that exceedence time calculations carried out by TSPROC need not be limited to time series which represent flow. A suitable time series could represent any environmental

quantity; the numbers following the FLOW keywords in the EXCEEDENCE_TIME block would then refer to the same quantity.

Use of the DELAY keyword requires special consideration. An EXCEEDENCE_TIME block in which this keyword is featured is shown below.

```
START EXCEEDENCE_TIME
  CONTEXT all
  SERIES_NAME sim_flow
  NEW_E_TABLE_NAME sim_extime
  EXCEEDENCE_TIME_UNITS days
  UNDER_OVER under
  FLOW 20.0
  DELAY 3.0
  FLOW 50.0
  DELAY 10.0
  FLOW 100.0
  DELAY 15.0
  FLOW 200.0
  DELAY 20.0
END EXCEEDENCE_TIME
```

An EXCEEDENCE_TIME block featuring the DELAY keyword.

If a DELAY keyword is used, it must directly follow the FLOW keyword to which it pertains. Furthermore a DELAY keyword must follow all FLOW keywords, or follow none at all.

Use of the DELAY keyword controls the way in which exceedence time is accumulated over the period spanned by a time series. In the example shown above, UNDER_OVER is set to "under". Hence, for the first FLOW entry (viz. 20), time over which elements of the "sim_flow" series are less than 20 is accumulated. However, for any one "below 20" event, time accumulation does not begin until 3 days after the beginning of the event; the time units pertaining to the DELAY keyword are assumed to be those supplied with the EXCEEDENCE_TIME_UNITS keyword. Thus the total exceedence time calculated by TSPROC for the flow of 20 will actually be the total time for which the flow was less than 20, but which was preceded by an interval of at least 3 days for which the flow was also less than 20.

Use of the DELAY keyword can be particularly useful when studying the effect of stream condition on biotic health. In many instances, the lethality of a particular adverse condition is a function of the magnitude of the condition and the duration over which the condition prevails. The more harmful the condition, the shorter the time which elapses before the condition exerts a deleterious influence on system health. This relationship is often described by "toxicity curves" relating, for example, concentration of a constituent to the exposure time. The greater is the concentration, the less is the exposure time required to cause damage.

By accumulating the time over which a user-specified chemical concentration or sediment load is exceeded (or for which flow is below a user-specified threshold), and by subtracting

the time required for the onset of harmful effects during each such "toxicity event", the total time over which biotic health suffered can be calculated. This may be an extremely useful model prediction, and one to which PEST's predictive analysis capabilities may be fruitfully turned.

If a user desires that EXCEEDENCE_TIME calculations be restricted to a certain date/time interval, a time series can be shortened prior to EXCEEDENCE_TIME calculations using the REDUCE_TIME_SPAN block.

GET_MUL_SERIES_GSFLOW_GAGE

The GET_MUL_SERIES_GSFLOW_GAGE block is used for extracting one or a number of time series from a "gage file" produced by the USGS GSFLOW model. This model produces two types of gage file, each with a slightly different header format. One of these lists model-calculated quantities pertaining to surface water features including lakes and rivers. The other lists model-calculated quantities computed by the UZ process. In either case, the file is comprised of multiple columns, each of which is associated with a header which specifies the information recorded in the column. One of these columns is elapsed simulation time. Data associated with any of the other columns can be extracted by providing the name of its header.

Keywords available in the GET_MUL_SERIES_GSFLOW_GAGE block are listed in the following table; an example of a GET_MUL_SERIES_GSFLOW_GAGE block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the GET_MUL_SERIES_GSFLOW_GAGE block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the GSFLOW gage file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
DATA_TYPE	*Mandatory*. The name of a data type to be imported; data types comprise column headers within a GSFLOW gage file. The name of a new series must immediately follow the DATA_TYPE keyword in the GET_MUL_SERIES_GSFLOW_GAGE block.	Any character string without internal spaces up to 30 characters in length.
NEW_SERIES_NAME	*Mandatory.* The name of a new series as stored by TSPROC. This keyword must immediately follow a DATA_TYPE keyword which specifies the dataset imported from the site sample file.	Any character string without internal spaces up to 10 characters in length.
MODEL_REFERENCE_DATE	*Mandatory.* The date corresponding to zero model simulation time. Simulation times are recorded in the GSFLOW gage file.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.

MODEL_REFERENCE_TIME	*Mandatory.* The time corresponding to zero model simulation time. Simulation times are recorded on the GSFLOW gage file.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_UNITS_PER_DAY	*Optional*. The number of time units as employed by the GSFLOW gage file comprising one day. If omitted, this is assumed to be 1.	A real number.
DATE_1	*Optional*. Terms of time any series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of any time series before this time on DATE_1 are not imported.	*hh:mm:ss*.
DATE_2	*Optional.* Terms of any time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of any time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords in a GET_MUL_SERIES_GSFLOW_GAGE block.

```
START GET_MUL_SERIES_GSFLOW_GAGE
  CONTEXT all
  FILE file1a.ggo
  DATA_TYPE flow
  NEW_SERIES_NAME flow_s
  DATA_TYPE uzf-runoff
  NEW_SERIES_NAME uzfr_s
  TIME_UNITS_PER_DAY  1
  MODEL_REFERENCE_DATE  1/1/2000
  MODEL_REFERENCE_TIME  12:00:00
  DATE_1 3/4/2005
  TIME_1 12:00:00
  DATE_2 6/9/2010
  TIME_2 00:00:00
END GET_ MUL_SERIES_GSFLOW_GAGE
```

A GET_MUL_SERIES_GSFLOW_GAGE block.

As for other TSPROC blocks which read multiple time series from the one file, entries pertaining to each imported series must be grouped. Thus one or a number of pairs of DATA_TYPE and NEW_SERIES_NAME keywords (in that order) must be provided in the GET_MUL_SERIES_GSFLOW_GAGE block. DATA_TYPE refers to a column header in the GSFLOW gage file, while NEW_SERIES_NAME provides the name of the imported series as stored by TSPROC. There is no limit to the number of these keyword pairs that can appear in a GET_MUL_SERIES_GSFLOW_GAGE block; a new series is imported for each

such pair. (Note that DATA_TYPE is case-insensitive; it will be matched to the pertinent heading in the GSFLOW gage file irrespective of the case employed for representation of this string either in the TSPROC input file, or in the GSFLOW gage file from which the series is imported.)

Each GSFLOW gage file contains a “time” column (with an appropriate header which designates it as such). Entries in this column are normally in days; indeed, unless a TIME_UNITS_PER_DAY keyword is present within the GET_MUL_SERIES_GFLOW_GAGE block, TSPROC will assume this to be the case. However where the time units are different from days, TSPROC must employ a time conversion factor as it imports the time series, this factor being supplied as the entry following the TIME_UNITS_PER_DAY keyword. Suppose that time units are actually hours; then TIME_UNITS_PER_DAY should be supplied as 24.0. Thus, as the name suggests, it is the number of time units employed by the model which collectively comprise one day.

To convert model simulation time to days and times, a reference date and time is needed, this being the date and time corresponding to a simulation time of zero. These must be supplied following the MODEL_REFERENCE_DATE and MODEL_REFERENCE_TIME keywords, both of which are mandatory in the GET_MUL_SERIES_GSFLOW_GAGE block.

The DATE_1, TIME_1 and DATE_2, TIME_2 keywords can be employed to restrict the length of the time series which is imported into TSPROC. No entries before DATE_1, TIME_1 or after DATE_2, TIME_2 are imported. Missing TIME_1 and TIME_2 entries denote a time of 00:00:00 in each case. Either or both of the DATE_1 and DATE_2 keywords can be omitted from the GET_MUL_SERIES_GSFLOW_GAGE block. If both of them are missing, the entirety of the time series is imported.

GET_MUL_SERIES_SSF

The role of the GET_MUL_SERIES_SSF block is very similar to that of the GET_SERIES_SSF block, the only difference being that multiple series can be imported from a site sample file in one operation using the former block, whereas only one series can be imported using the latter block. See Appendix B for the format of a site sample file. Because a site sample file is used for time series storage by other members of the PEST Surface Water Utilities suite, TSPROC can readily import time series data written by other members of the suite. Also, if it is desired that TSPROC be used in the calibration of a model for which it is presently incapable of directly importing results, then this can be implemented by writing a small translation program which converts the outputs of that model to site sample file format. This program would be run between the model and TSPROC as part of a composite model calibrated by PEST.

The table below shows the keywords appearing in a GET_MUL_SERIES_SSF block. An example of a GET_MUL_SERIES_SSF block is shown following that. Note that the CONTEXT keyword(s) must precede all other keywords cited in this block.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the GET_MUL_SERIES_SSF block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the site sample file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
SITE	*Mandatory*. The name of a site within the site sample file for which a time series is to be imported. The name of a new series must immediately follow the SITE keyword in the GET_MUL_SERIES_SSF block.	Any character string without internal spaces up to 10 characters in length.
NEW_SERIES_NAME	*Mandatory.* The name of a new series as stored by TSPROC. This keyword must immediately follow the SITE keyword which specifies the dataset imported from the site sample file.	Any character string without internal spaces up to 10 characters in length.
DATE_1	*Optional*. Terms of time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.

TIME_1	*Optional*. Terms of time series before this time on DATE_1 are not imported.	*hh:mm:ss*.
DATE_2	*Optional.* Terms of time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords in a GET_MUL_SERIES_SSF block.

```
START GET_MUL_SERIES_SSF
  CONTEXT all
  FILE flows.smp
  SITE rebec_ck
  NEW_SERIES_NAME rebecca
  SITE horton_ck
  NEW_SERIES_NAME horton
  SITE sandy_ck
  NEW_SERIES_NAME sandy
  DATE_1 06/03/1970
  TIME_1 12:00:00
  DATE_2 09/01/1980
  TIME_1 00:00:00
END GET_MUL_SERIES_SSF
```

A GET_MUL_SERIES_SSF block.

The DATE_ and TIME_ specifiers are optional. If they are omitted then the entire time series pertaining to each of the nominated sites is imported. If a DATE_1 keyword is present but a TIME_1 keyword is absent, then TIME_1 is assumed to be 00:00:00; similarly for DATE_2. If TIME_1 is present then DATE_1 must be present; the same holds for TIME_2.

A GET_MUL_SERIES_SSF block can contain multiple incidences of the SITE and NEW_SERIES_NAME keywords. However these must be supplied in pairs with the SITE keyword immediately preceding the pertinent NEW_SERIES_NAME keyword. The character string associated with the SITE keyword, must pertain to a site which appears within the nominated site sample file; The same site cannot be supplied twice.

Correct operation of the instructions contained within the GET_MUL_SERIES_SSF block assumes that the site sample file read using this block is correct and consistent. The integrity of a site sample file can be checked with the utility program SMPCHEK supplied with the Surface Water Utility suite.

GET_MUL_SERIES_STATVAR

"STATVAR files" are written by the Precipitation-Runoff Modelling System (PRMS) model, as well as by the Modular Modelling System (MMS) model. An example of such a file is provided below.

```
4
node_cfs.musroute 132
runoff.obs 16
node_cfs.musroute 87
node_cfs.musroute 14
1 1975 6 1 0 0 0 2.490942 6.434562 3.300000 0.000000
2 1975 6 2 0 0 0 2.501948 7.389743 2.800000 0.000000
3 1975 6 3 0 0 0 2.476184 9.652343 2.900000 0.000000
etc
```

Part of a STATVAR file.

The STATVAR file begins with the number of series *N* represented in the file. Following that are *N* lines, each containing a variable name followed by a "location identifier". Taken together, these uniquely identify a series. Following these *N* lines are the data comprising the time series themselves. The first entry on each such line is the model simulation day. Following that are the year, month, day, hour, minute and second respectively corresponding to series entries on that line followed by the entries themselves; entries are in the same order as variable name and location id entries provided in the header to the file.

Keywords that can be employed in a GET_MUL_SERIES_STATVAR block are provided in the table below. An example of a GET_MUL_SERIES_STATVAR block follows that.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the GET_MUL_SERIES_STATVAR block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the STATVAR file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
VARIABLE_NAME	*Mandatory*. The name of a variable within the STATVAR file for which a time series is to be imported. Collectively a variable name and location id denote a unique time series.	Any character string without internal spaces up to 50 characters in length.

LOCATION_ID	*Mandatory*. The location identifier for an imported time series. The location id together with a variable name collectively denote a unique time series. This keyword must immediately follow a VARIABLE_NAME keyword.	An integer.
NEW_SERIES_NAME	*Mandatory.* The name of a new series as stored by TSPROC. This keyword must immediately follow a LOCATION_ID keyword.	Any character string without internal spaces up to 10 characters in length.
DATE_1	*Optional*. Terms of any time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of any time series before this time on DATE_1 are not imported.	*hh:mm:ss*.
DATE_2	*Optional.* Terms of any time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of any time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords in a GET_MUL_SERIES_STATVAR block.

```
START GET_MUL_SERIES_STATVAR
  CONTEXT all
  FILE statvar.dat
  VARIABLE_NAME node_cfs.musroute
  LOCATION_ID 132
  NEW_SERIES_NAME runoff16
  VARIABLE_NAME node_cfs.musroute
  LOCATION_ID  2
  NEW_SERIES_NAME cfs2
  DATE_1 6/4/1975
  TIME_1 12:00:00
  DATE_2 10/29/2001
  TIME_2 00:00:00
END GET_ MUL_SERIES_STATVAR
```

A GET_MUL_SERIES_STATVAR block.

A GET_MUL_SERIES_STATVAR block can be used to import one or a number of series from a STATVAR file. Each imported series is identified by a VARIABLE_NAME and LOCATION_ID. These two keywords must be supplied in that order, immediately followed by a NEW_SERIES_NAME keyword for each series to be imported. As many such triplets must be featured in this block as there are series to import.

The DATE_1, TIME_1, DATE_2 and TIME_2 keywords are optional. If none of these is supplied then the entirety of each time series is imported. If any of these are present, then no series terms which precede DATE_1, TIME_1 or postdate DATE_2, TIME_2 will be imported. If TIME_1 or TIME_2 is omitted, a time of 00:00:00 is assumed in either case.

GET_SERIES_PLOTGEN

The GET_SERIES_PLOTGEN block governs importation of time series data from a HSPF PLOTGEN file into TSPROC. There is an important difference between use of this block and use of some of the other blocks which import time series data into TSPROC; when importing data from a PLOTGEN file, more than one time series can be imported using the same block. This saves TSPROC from having to read a HSPF PLOTGEN file many times in order to import multiple time series produced during a HSPF run.

Another slight difference between series importation using the GET_SERIES_PLOTGEN block and series importation using other TSPROC blocks is that the ordering of some keywords is important in the GET_SERIES_PLOTGEN block. In particular, each NEW_SERIES_NAME keyword provided in this block must directly follow a LABEL keyword so that the association between the time series label in the HSPF PLOTGEN file and the name of the new series as stored within TSPROC is clear. Due to the multiple time series importation capabilities of the GET_SERIES_PLOTGEN block, more than one of these LABEL/NEW_SERIES_NAME pairs can be present within any such block. See the table below.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the GET_SERIES_PLOTGEN block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the PLOTGEN file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains spaces.
LABEL	*At least one LABEL keyword must be present*. This is the PLOTGEN label pertaining to a time series which is to be imported.	Any character string up to 20 characters in length. If the string contains blank characters, enclose it in quotes.
NEW_SERIES_NAME	*Mandatory.* The name of the new series as stored by TSPROC. This must immediately follow the LABEL keyword pertaining to the imported time series.	Any character string without spaces up to 10 characters in length.
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yyyy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not imported.	*hh:mm:ss*

DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords in a GET_SERIES_PLOTGEN block.

```
START GET_SERIES_PLOTGEN
  CONTEXT all
  FILE hspfout.plt
  LABEL "total outflow"
  NEW_SERIES_NAME t_outflow
  LABEL interflow
  NEW_SERIES_NAME interflow
  DATE_1 6/1/1976
  TIME_1 00:12:00
  DATE_2 7/1/1976
  TIME_2 00:12:00
END GET_SERIES_PLOTGEN
```

A GET_SERIES_PLOTGEN block.

DATE_ and TIME_ specifiers are optional in a GET_SERIES_PLOTGEN block. If they are absent from the block, then the entire time series pertaining to each nominated label is imported. If a DATE_1 keyword is present but a TIME_1 keyword is absent, then TIME_1 is assumed to be 00:00:00; the same applies for DATE_2. However if TIME_1 is present then DATE_1 must also be present; the same holds for TIME_2.

GET_SERIES_SSF

Instructions provided in a GET_SERIES_SSF block allow TSPROC to import a time series from a site sample file; see Appendix B for the format of this file. Because a site sample file is used for time series storage by other members of the PEST Surface Water Utilities suite, TSPROC can readily import time series data written by other members of the suite. Also, if it is desired that TSPROC be used in the calibration of a model for which it is presently incapable of directly importing results, then this can be implemented by writing a small translation program which converts the outputs of that model to site sample file format. This program would be run between the model and TSPROC as part of a composite model calibrated by PEST.

The table below shows the keywords pertaining to a GET_SERIES_SSF block. An example of a GET_SERIES_SSF block is shown following that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the GET_SERIES_SSF block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the site sample file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
SITE	*Mandatory*. The name of the site within the site sample file for which a time series is to be imported.	Any character string without internal spaces up to 10 characters in length.
NEW_SERIES_NAME	*Mandatory.* The name of the new series as stored by TSPROC.	Any character string without internal spaces up to 10 characters in length.
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not imported.	*hh:mm:ss*.
DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.

TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords in a GET_SERIES_SSF block.

```
START GET_SERIES_SSF
  CONTEXT all
  FILE flows.smp
  SITE rebec_ck
  NEW_SERIES_NAME rebecca
  DATE_1 06/03/1970
  TIME_1 12:00:00
  DATE_2 09/01/1980
  TIME_1 00:00:00
END GET_SERIES_SSF
```

A GET_SERIES_SSF block.

The DATE_ and TIME_ specifiers are optional. If they are omitted then the entire time series pertaining to the nominated site is imported. If a DATE_1 keyword is present but a TIME_1 keyword is absent, then TIME_1 is assumed to be 00:00:00; similarly for DATE_2. If TIME_1 is present then DATE_1 must be present; the same holds for TIME_2.

GET_SERIES_TETRAD

A GET_SERIES_TETRAD block reads data from a TETRAD plot file. As presently programmed it is assumed that the plot file contains well data only (ie. that the TETRAD IPLWA variable has been set to zero.) Multiple time series can be read from this file, each such series being assigned to a particular well/object combination.

The table below shows the keywords pertaining to a GET_SERIES_TETRAD block. An example of a GET_SERIES_TETRAD block is shown following that. Note that the CONTEXT keyword(s) which must precede all other keywords occurring in this block.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the GET_SERIES_TETRAD block will be processed.	Any character string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the TETRAD plot file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
WELL_NAME	*Mandatory*. The name of a well within the TETRAD plot file. This, in combination with the OBJECT_NAME keyword, defines the series to be imported.	Any character string without internal spaces up to 25 characters in length.
OBJECT_NAME	*Mandatory*. The name of an object for which data is recorded in the TETRAD plot file. This, in combination with the WELL_NAME keyword, defines the series to be imported. This keyword must immediately follow a WELL_NAME keyword.	Any character string without internal spaces up to 25 characters in length.
NEW_SERIES_NAME	*Mandatory.* The name of the new series as stored by TSPROC. This keyword must immediately follow an OBJECT_NAME keyword.	Any character string without internal spaces up to 10 characters in length.
MODEL_REFERENCE_DATE	*Mandatory.* The date pertaining to zero simulation time. Simulation times are recorded by TETRAD on its plot output file.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.

MODEL_REFERENCE_TIME	*Mandatory.* The time pertaining to zero simulation time. Simulation times are recorded by TETRAD on its plot output file.	*hh:mm:ss*.
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not imported.	*hh:mm:ss*.
DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords in a GET_SERIES_TETRAD block.

```
START GET_SERIES_TETRAD
  CONTEXT all
  FILE dv1.plt
  MODEL_REFERENCE_DATE 1/1/2000
  MODEL_REFERENCE_TIME 00:00:00
  WELL_NAME 2733
  OBJECT_NAME pav
  NEW_SERIES_NAME pav2733
  WELL_NAME 6518p
  OBJECT_NAME qtotenth
  NEW_SERIES_NAME qth6518
END GET_SERIES_TETRAD
```

A GET_SERIES_TETRAD block.

There is no limit to the number of time series which can be imported from a TETRAD plot file (except as imposed by the storage limits of TSPROC itself). Three keywords within a GET_SERIES_TETRAD block collectively define a series to be imported and the name to be assigned to this series for the purposes of further TSPROC processing. These are the WELL_NAME, OBJECT_NAME and NEW_SERIES_NAME keywords. To avoid confusion in series definition, these must be listed consecutively within a GET_SERIES_TETRAD block for each time series which is to be imported.

As in other TSPROC blocks which implement series importation, the user is able to limit the length of the imported time series by providing a time window using the optional DATE_1, TIME_1, DATE_2 and TIME_2 keywords. The MODEL_REFERENCE_DATE and MODEL_REFERENCE_TIME are mandatory; these relate zero simulation time to a specific date and time.

GET_SERIES_UFORE_HYDRO

The GET_SERIES_UFORE_HYDRO block allows a user to extract a time series from a file which serves as an input or output file for the UFORE-HYDRO urban forest effects model. An example of such a file follows.

```
13
2.99111406e-005
1.8395021e-005
1.72083527e-005
1.66953228e-005
1.63247823e-005
1.60653292e-005
1.58376155e-005
1.56827719e-005
1.55282626e-005
1.53740841e-005
1.52726813e-005
1.52230862e-005
1.51738124e-005
```

An example UFORE-HYDRO input/output file.

The first line of a UFORE-HYDRO time series file contains a single integer indicating the number of entries to follow. Then follow the time series entries themselves, with one entry on each line. Entries correspond to equal elapsed time intervals.

Keywords found in a GET_SERIES_UFORE_HYDRO block are listed in the table below; an example follows that.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the context strings is "all", the GET_SERIES_UFORE_HYDRO block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the UFORE-HYDRO file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series as stored by TSPROC.	Any character string without internal spaces up to 10 characters in length.
TIME_INCREMENT	*Mandatory*. The time increment corresponding to successive elements in the imported time series.	An integer representing time in seconds.

MODEL_REFERENCE_ DATE	*Mandatory*. The date on which the model simulation is assumed to begin.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
MODEL_REFERENCE_ TIME	*Mandatory*. The time at which the model simulation is assumed to begin.	*hh:mm:ss*
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not imported.	*hh:mm:ss*
DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not imported.	*hh:mm:ss*

Keywords within a GET_SERIES_UFORE_HYDRO block.

```
START GET_UFORE_HYDRO
  CONTEXT all
  FILE totalqq.dat
  MODEL_REFERENCE_DATE 3/3/2004
  MODEL_REFERENCE_TIME 12:00:00
  NEW_SERIES_NAME flow
  TIME_INCREMENT 3600
END GET_SERIES_UFORE_HYDRO
```

Example of a GET_SERIES_UFORE_HYDRO block.

It is important that the following be noted.

1. The TIME_INCREMENT must be supplied in seconds (and as an integer).
2. The first time series element is assumed to occur one TIME_INCREMENT after the date/time given by the MODEL_REFERENCE_DATE and MODEL_REFERENCE_TIME.

GET_SERIES_WDM

Instructions provided in this block allow TSPROC to import a time series from a Watershed Data Management (WDM) file. Many hydrologic and water-quality models and analyses developed by the U.S. Geological Survey and the U.S. Environmental Protection Agency currently use a WDM file. The WDM file is a binary file which provides the user with a common data base for many applications, thus eliminating the need to reformat data from one application to another.

The table below shows the keywords permissible in a GET_SERIES_WDM block. An example of a GET_SERIES_WDM block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the context strings is "all", the GET_SERIES_WDM block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory.* The name of the WDM file containing the time series to be imported.	Any file name up to 120 characters in length. Use quotes if the filename contains blank characters.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series as stored by TSPROC.	Any character string without internal spaces up to 10 characters in length.
DSN	*Mandatory*. The data set number of the time series to be imported.	Any integer for which a time series dataset is available within the nominated WDM file.
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not imported.	*hh:mm:ss*
DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not imported.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not imported.	*hh:mm:ss*

DEF_TIME	*Optional*. If the time series imported from a WDM file has a time step of a day or greater, each term pertains to a date, but no time. Upon importation into TSPROC, each term of such a series is referenced to REF_TIME. Default is 00:00:00.	*hh:mm:ss* Note that if DEF_TIME is supplied as 24:00:00 a time of 00:00:00 on the following day will be assigned to all samples.
FILTER	*Optional*. Terms of the time series which have this value are ignored upon importation into TSPROC.	Real number.

Keywords within a GET_SERIES_WDM block.

```
START GET_SERIES_WDM
  CONTEXT all
  FILE catchment.wdm
  DSN 1013
  NEW_SERIES_NAME coal_ck
  DATE_1 06/03/1970
  TIME_1 12:00:00
  DATE_2 09/01/1980
  TIME_1 00:00:00
  DEF_TIME 12:00:00
  FILTER -999.99
END GET_SERIES_WDM
```

A GET_SERIES_WDM block.

The DATE_ and TIME_ specifiers are optional in a GET_SERIES_WDM block. If they are omitted then the entire time series pertaining to the nominated data set number is imported. If a DATE_1 keyword is present but a TIME_1 keyword is absent, then TIME_1 is assumed to be 00:00:00; similarly for DATE_2. If TIME_1 is present then DATE_1 must be present; the same holds for TIME_2.

If the sample interval for a time series stored in a WDM file is a day or greater, then each term of the series will have no time reference; however within TSPROC each time series term is associated with both a date *and* a time. When importing such a time series into TSPROC, TSPROC's default behaviour is to assign each term a time of 00:00:00 on the day with which it is associated. However, this time can be altered to the user's choice using the optional DEF_TIME keyword. Note that if DEF_TIME is supplied as "24:00:00" then each sample will be assigned a time of 00:00:00 on the following day.

LIST_OUTPUT

The LIST_OUTPUT block provides the means whereby the outcomes of calculations carried out by TSPROC can be written to an ASCII (ie. text) file. The format of this file is such that these quantities can be easily read by a user. They can also be easily read by PEST. An instruction file by which PEST can read the contents of a LIST_OUTPUT file can be generated automatically using the WRITE_PEST_FILES block.

Keywords associated with a LIST_OUTPUT block are recorded in the following table. An example of a LIST_OUTPUT block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the LIST_OUTPUT block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
FILE	*Mandatory*. The name of the file to be written by the LIST_OUTPUT block.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
SERIES_NAME	*Optional.* The name of a time series to be written by the LIST_OUTPUT block to its output file.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
SERIES_FORMAT	*Mandatory if a SERIES_NAME keyword is present.* Determines whether dates, times and the series name should accompany series terms in the file generated by the LIST_OUTPUT block.	"short" or "long"
C_TABLE_NAME	*Optional*. The name of a c_table to be written by the LIST_OUTPUT block to its output file.	A name of 10 characters or less in length referencing a c_table stored within TSPROC's memory.
S_TABLE_NAME	*Optional*. The name of an s_table to be written by the LIST_OUTPUT block to its output file.	A name of 10 characters or less in length referencing an s_table stored within TSPROC's memory.

V_TABLE_NAME	*Optional*. The name of a v_table to be written by the LIST_OUTPUT block to its output file.	A name of 10 characters or less in length referencing a v_table stored within TSPROC's memory.
E_TABLE_NAME	*Optional*. The name of an e_table to be written by the LIST_OUTPUT block to its output file.	A name of 10 characters or less in length referencing an e_table stored within TSPROC's memory.

Keywords in a LIST_OUTPUT block.

```
START LIST_OUTPUT
  CONTEXT all
  FILE output.txt
  SERIES_NAME flow_216
  SERIES_NAME flow_342
  V_TABLE_NAME vol_216
  V_TABLE_NAME vol_342
  S_TABLE_NAME st_216
  S_TABLE_NAME st_342
  E_TABLE_NAME dur_216
  E_TABLE_NAME dur_342
  C_TABLE_NAME comp_ser
  SERIES_FORMAT short
END LIST_OUTPUT
```

A LIST_OUTPUT block.

Any number of time series, c_tables, s_tables, v_tables and e_tables can be written to a file generated by the LIST_OUTPUT block. Hence as many of the keywords pertaining to these entities as desired can be supplied in this block. In generating its output files, time series are written first, followed by s_tables, followed by c_tables, followed by v_tables, and finally e_tables. However the ordering of the individual entities of each type within the different segments of the TSPROC output file is the same as the order in which respective keywords referencing those entities are supplied in the LIST_OUTPUT block.

If a SERIES_NAME keyword is provided in a LIST_OUTPUT block then a SERIES_FORMAT keyword must also be provided; options are "short" and "long". If the former option is supplied, the LIST_OUTPUT block will list the terms of the time series as a single column in its output file. If the latter option is supplied the terms of the time series will be accompanied by the date and time corresponding to the term, as well as the name of the time series. This format corresponds to that of a site sample file (see Appendix B) and can thus be used by other members of the PEST Surface Water Utilities; note however that the header to each time series, written by the LIST_OUTPUT block to its output file, must first be removed.

If you are running TSPROC as part of a composite model under the control of PEST, it is best to use the "short" option for time series formatting. This is because, where a time series is large, a considerable amount of computation time may be spent in converting TSPROC's

internal representation of sample dates and times to the *dd/mm/yyyy* (or *mm/dd/yyyy*) and *hh:mm:ss* formats required for output listing. This can add considerably to overall composite model execution time. Note also that if the "long" protocol is employed, in accordance with site sample file protocol, TSPROC does not represent midnight as "24:00:00"; instead midnight is represented as 00:00:00 on the following day.

Output formatting for other TSPROC entities is such that they are clearly labelled and easily understood by the user. In the case of s_tables and c_tables, it is important to note that statistics not requested in the SERIES_STATISTICS or SERIES_COMPARE block in which an s_table or c_table respectively is created are not recorded in the file written by the LIST_OUTPUT block. Thus if this file is considered as the output file of a composite model, and that composite model is being calibrated by PEST, such statistics will not be included in the calibration process.

Exceedence times stored in an e_table are recorded by the LIST_OUTPUT block both as accumulated times, and as proportions of the total time spanned by the parent time series. Note that if this file is used by PEST, only the latter quantities (ie. the exceedence *proportions*) are actually read by PEST on the basis of the instruction file created through a WRITE_PEST_FILES block.

NEW_SERIES_UNIFORM

The NEW_SERIES_UNIFORM block creates a uniform-valued time series with series entries placed at uniform (or almost uniform) time intervals. Keywords belonging to the NEW_SERIES_UNIFORM block are listed in the following table. An example NEW_SERIES_UNIFORM block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the NEW_SERIES_UNIFORM block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
NEW_SERIES_NAME	*Mandatory.* The name of the new, uniform-valued time series.	Any character string without internal spaces up to 10 characters in length.
DATE_1	*Mandatory*. The starting date of the new time series..	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT in the SETTINGS block.
TIME_1	*Mandatory*. The starting time of the new time series..	*hh:mm:ss*
DATE_2	*Mandatory.* Terms of the new time series are not created after this date.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT in the SETTINGS block.
TIME_2	*Mandatory*. Terms of the new time series are not created after this time on DATE_2.	*hh:mm:ss*
TIME_INTERVAL	*Mandatory.* The number of TIME_UNITS (see below) between terms of the new series.	An integer greater than zero.
TIME_UNIT	*Mandatory.* The time units in which the TIME_INTERVAL is expressed.	"Seconds", "minutes", "hours", "days", "months" or "years".
NEW_SERIES_VALUE	*Mandatory.* The value supplied to all new terms in the new time series.	A real number.

Keywords in a NEW_SERIES_UNIFORM block.

```
START NEW_SERIES_UNIFORM
  CONTEXT all
  TIME_INTERVAL 2
  TIME_UNIT  days
  DATE_1  1/1/2000
  TIME_1 12:00:00
  DATE_2  15/12/2005
  TIME_2 13:00:00
  NEW_SERIES_NAME  series1
  NEW_SERIES_VALUE  5.0
END NEW_SERIES_UNIFORM
```

Example of a NEW_SERIES_UNIFORM block.

The behaviour of the NEW_SERIES_UNIFORM block is slightly different depending on whether TIME_UNIT is supplied as “seconds”, “minutes”, “hours” or “days” on the one hand, or “months” or “years” on the other hand. In the former case the series time interval is strictly TIME_INTERVAL times the TIME_UNIT; thus the time increment between successive terms of the series is strictly uniform. For example, if the TIME_INTERVAL is supplied as 2 and the TIME_UNIT is supplied as “hours”, then all samples are 2 hours apart. The first sample occurs on DATE_1, TIME_1; the last sample is no later than DATE_2, TIME_2. (Note that, in contrast to most other blocks, all of the DATE_1, TIME_1, DATE_2 and TIME_2 keywords must be supplied.)

On the other hand if TIME_UNIT is set to “months”, then all terms of the new series occur on the same day of the month, and at the same time as the initial time TIME_1. Thus terms are variously 28, 29, 30 or 31 days apart. (In this case TSPROC will reject a DATE_1 in which the date is 29th, 30th or 31st of the month.) If TIME_UNIT is set to “years”, then all terms of the new time series occur on the same date, but separated by TIME_INTERVAL years. In this case TSPROC will not allow DATE_1 to be 29th February.

All terms of the new series are assigned the same value, this being the user-supplied NEW_SERIES_VALUE.

The NEW_TIME_SERIES block can be useful as a precursor to digital filtering. As explained in documentation to the DIGITAL_FILTER block, digital filtering can only take place on a time series for which the terms are separated by a constant time increment. If filtering must be performed on an observed time series that has not been sampled at such a constant increment, the latter can be interpolated to a constant time base series (created using the NEW_SERIES_UNIFORM block) using the NEW_TIME_BASE block. Filtering can then be undertaken on the interpolated time series.

NEW_TIME_BASE

The NEW_TIME_BASE block is used to carry out time-interpolation from the sample times pertaining to one time series to the sample times pertaining to another. Keywords belonging to the NEW_TIME_BASE block are listed in the following table. An example NEW_TIME_BASE block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the NEW_TIME_BASE block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series whose terms are to be time-interpolated to a new time base.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series produced as an outcome of time-interpolation.	Any character string without internal spaces up to 10 characters in length.
TB_SERIES_NAME	*Mandatory*. The name of the time series to whose dates and times time-interpolation will take place.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.

Keywords in a NEW_TIME_BASE block.

```
START NEW_TIME_BASE
  CONTEXT all
  SERIES_NAME mod_flow
  TB_SERIES_NAME obs_flow
  NEW_SERIES_NAME int_flow
END NEW_TIME_BASE
```

Example of a NEW_TIME_BASE block.

Time interpolation of one time series to the time-base of another will only occur if the time-span of the latter time series is equal to, or smaller than, that of the former time series. The result will be a new time series with terms pertaining to exactly the same dates and times as those of the time-base time series. If the original time series and the time-base time series pertain to the same data type, this will allow the two series to be directly compared with each

other. Such a comparison of "apples with apples" is crucial when calibrating a model against field data. Hence one of the principal roles of TSPROC when used as a model post-processor in a "composite model" run by PEST, is to carry out this all-important time-interpolation of model-generated time series to the dates and times of their measured counterparts. An interpolated time series produced in this manner can then be written to a TSPROC output file (using the LIST-OUTPUT block), where it can be read by PEST and compared with measured values recorded in a PEST control file. Both the PEST control file, and the instruction file by which the time-interpolated time series can be read from the LIST_OUTPUT file, can be written using the WRITE_PEST_FILES block.

REDUCE_TIME_SPAN

The REDUCE_TIME_SPAN block reduces the time spanned by a time series. This may be a useful precursor to other aspects of TSPROC processing. For example, using the REDUCE_TIME_SPAN block, the time spanned by an "observed time series" can be reduced to that spanned by a model-generated time series. This will allow time interpolation from the model's output times to the times at which measurements were made, to be carried out using the NEW_TIME_BASE block.

Keywords found in a REDUCE_TIME_SPAN block are listed in the table below. An example of a REDUCE_TIME_SPAN block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the REDUCE_TIME_SPAN block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series whose time span is to be reduced.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series produced as an outcome of time span reduction.	Any character string without internal spaces up to 10 characters in length.
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not copied to the new time series.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not copied to the new time series.	*hh:mm:ss*
DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not copied to the new time series.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT in the SETTINGS block.
TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not copied to the new time series.	*hh:mm:ss*

Keywords in a REDUCE_TIME_SPAN block.

```
START REDUCE_TIME_SPAN
  CONTEXT all
  SERIES_NAME intflow
  NEW_SERIES_NAME  intflow_1
  DATE_1 02/01/1976
  TIME_1 13:13:00
  DATE_2 06/01/1976
  TIME_2 00:00:00
END REDUCE_TIME_SPAN
```

A REDUCE_TIME_SPAN block.

When a new time series is created by reducing the time span of an existing time series, the original time series still remains within TSPROC's memory. If desired, it can be removed using the ERASE_ENTITY block.

At least one of DATE_1 or DATE_2 must be supplied. If the corresponding TIME_ keyword is not supplied, a default time of 00:00:00 is used. If the DATE_1 keyword is omitted DATE_1 and TIME_1 are assumed to be the first date and time cited in the original time series, ie. no time-span reduction from the front of the time series takes place. Similarly, if the DATE_2 keyword is omitted, no time-span reduction takes place from the end of the existing time series. Note that a TIME_ keyword cannot be supplied without the corresponding DATE_ keyword.

SERIES_BASE_LEVEL

Use of the SERIES_BASE_LEVEL block allows a user to subtract a constant amount from all terms of a time series. This constant amount is the value of one term of an existing time series, either the time series from which subtraction is taking place, or another time series stored within the memory of TSPROC.

A common use of the SERIES_BASE_LEVEL block is in calculation of changes in the quantity represented by the time series over the data recording or model simulation interval that gave rise to the time series in the first place. In this case the first term of the time series may be taken as the base level, this term being subtracted from all other elements of the time series to create the new series with altered base level. SERIES_BASE_LEVEL functionality allows this new series to either replace the original time series or to exist as its own separate entity.

Keywords found in a SERIES_BASE_LEVEL block are listed in the table below. An example of a SERIES_BASE_LEVEL block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the SERIES_BASE_LEVEL block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series whose base level is to be altered.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
SUBSTITUTE	*Mandatory.* If this is supplied as "yes", the new time series replaces the old one in TSPROC's memory and retains the same name. If it is supplied as "no", a new series is created.	"yes" or "no".
NEGATE	*Optional.* If this is supplied as "yes" all terms of the new base series are multiplied by –1 after subtraction of the constant.	yes" or "no".
NEW_SERIES_NAME	*Mandatory if SUBSTITUTE is supplied as "yes".* The name of the new time series produced as an outcome of base level alteration.	Any character string without internal spaces up to 10 characters in length.

BASE_LEVEL_SERIES_NAME	*Mandatory*. The name of the time series of which one element will be subtracted from all elements of the original time series to effect the base level change.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
BASE_LEVEL_DATE	*Mandatory*. This is used in conjunction with BASE_LEVEL_TIME to identify the term of series BASE_LEVEL_SERIES_NAME which is subtracted from all elements of SERIES to effect the base level change.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT in the SETTINGS block.
BASE_LEVEL_TIME	*Mandatory.* This is used in conjunction with BASE_LEVEL_DATE to identify the term of series BASE_LEVEL_SERIES_NAME which is subtracted from all elements of SERIES to effect the base level change.	*hh:mm:ss*

Keywords in a SERIES_BASE_LEVEL block.

```
START SERIES_BASE_LEVEL
  CONTEXT all
  SERIES_NAME head
  BASE_LEVEL_SERIES_NAME head
  BASE_LEVEL_DATE 01/04/1996
  BASE_LEVEL_TIME 12:00:00
  SUBSTITUTE no
  NEGATE yes
  NEW SERIES_NAME drawdown
END SERIES_BASE_LEVEL
```

A SERIES_BASE_LEVEL block.

As is documented elsewhere in this manual, the SERIES_EQUATION block can also be used to subtract a constant from the terms of a series. However in that case, the constant is supplied as a number in an equation. In the case of the SERIES_BASE_LEVEL block, the subtractor is a term in a series, identified through the name of the series and the date and time to which the term pertains. If there is no term corresponding to the supplied date and time, TSPROC will cease execution with an appropriate error message.

The NEGATE keyword can be useful in incidences such as where it is desired that drawdown be calculated from head. Drawdown is calculated as the negative of the change in head from its initial value. Thus after base level alteration by subtraction of the initial series term, all terms of the new time series are multiplied by –1.

SERIES_CLEAN

Using the SERIES_CLEAN block, unwanted terms can be eliminated from a time series or replaced with a preferred value. This is sometimes required for correcting the deleterious effects of model misbehaviour whereby model-generated time-series are "polluted" with intermittent spurious values. It can also be used for eliminating outliers in an observation time series.

Keywords pertaining to the SERIES_CLEAN block are listed in the table below. An example SERIES_CLEAN block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the SERIES_CLEAN block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series whose terms are to be "cleaned".	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_SERIES_NAME	*Mandatory if SUBSTITUTE_VALUE is "delete"; optional otherwise* . The name of a new time series formed by removal or replacement of terms in the original time series.	Any character string without internal spaces up to 10 characters in length.
LOWER_ERASE_BOUNDARY	*Optional*. Terms equal to and above this threshold and equal to and below the UPPER_ERASE_BOUNDARY threshold are removed or replaced.	A real number.
UPPER_ERASE_BOUNDARY	*Optional*. Terms equal to or below this threshold and equal to or above the LOWER_ERASE_BOUNDARY threshold are removed or replaced.	A real number.
SUBSTITUTE_VALUE	*Mandatory*. If supplied as a real number this is the value substituted for terms between the upper and lower erase thresholds. If supplied as "delete", instructs TSPROC to delete terms between these thresholds.	A real number or "delete".

Keywords within a SERIES_CLEAN block.

```
START SERIES_CLEAN
  CONTEXT all
  SERIES_NAME series1
  LOWER_ERASE_BOUNDARY  100.0
  UPPER_ERASE_BOUNDARY 200.0
  SUBSTITUTE_VALUE delete
  NEW_SERIES_NAME series2
END SERIES_CLEAN
```

A SERIES_CLEAN block.

The SERIES_CLEAN block presents the user with a number of different options for handling unwanted terms. In the simplest case these terms are replaced by the number supplied through the SUBSTITUTE_VALUE keyword. If this is done, terms can be replaced "in situ" (ie. in the existing time series without creating a new one), or a new time series can be created to hold the altered time series, with the original time series remaining intact. If a NEW_SERIES_NAME keyword is supplied, the latter option is taken; if not, the former option is taken.

A further option is for unwanted terms to be eradicated altogether. This is achieved by supplying the string "delete" with the SUBSTITUTE_VALUE keyword instead of a real number. In this case TSPROC insists that a NEW_SERIES_NAME keyword be supplied in the SERIES_CLEAN block, for the altered time series will be stored as a new entity, leaving the original one intact; the latter can then be erased if desired using the ERASE_ENTITY block.

Terms of a series are identified for deletion or replacement using the LOWER_ERASE_BOUNDARY and UPPER_ERASE_BOUNDARY keywords. Either one or both of these keywords can be supplied. If both of them are supplied, all terms of the time series between and including the specified boundary values are replaced or deleted. If only the LOWER_ERASE_BOUNDARY keyword is supplied, all terms equal to and above this threshold are removed or replaced; if only the UPPER_ERASE_BOUNDARY keyword is supplied, all terms equal to or below this boundary are removed or replaced. (If you are in any doubt of the action of the SERIES_CLEAN block when only one of these keywords is supplied, then supply both of them, with one of them either very high or very low. However if you do this, note that TSPROC will not accept numbers whose absolute value is greater than about 10^{37}.

SERIES_COMPARE

The SERIES_COMPARE block calculates statistics that quantify the similarity of one time series with another. The outcomes of these calculations are placed in a c_table (which can be written to a file using the LIST_OUTPUT block). Keywords pertaining to the SERIES_COMPARE block are listed in the table below. An example SERIES_COMPARE block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the SERIES_COMPARE block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME_SIM	*Mandatory.* The name of the "simulated" time series whose terms are to be compared with the "observed" time series.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
SERIES_NAME_OBS	*Mandatory.* The name of the "observed" time series whose terms are to be compared with the "simulated" time series.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
SERIES_NAME_BASE	*Optional if either the COEFFICIENT_OF_EFFICIENCY or INDEX_OF_AGREEMENT keyword is present.* The name of a baseline time series that can be used in the calculation of these two quantities.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_C_TABLE_NAME	*Mandatory.* The name of the new c_table used to store the outcomes of comparison statistics calculations.	Any character string without internal spaces up to 10 characters in length.
BIAS	*Optional*. Requests calculation of bias between the observed and simulated time series (*B* in the equations below).	"yes" or "no". Default is "no".
STANDARD_ERROR	*Optional*. Requests calculation of the standard error between the observed and simulated time series (*S* in the equations below).	"yes" or "no". Default is "no".

RELATIVE_BIAS	*Optional*. Requests calculation of the relative bias between the observed and simulated time series (B_r in the equations below).	"yes" or "no". Default is "no".
RELATIVE_STANDARD_ERROR	*Optional*. Requests calculation of the relative standard error between the observed and simulated time series (S_r in the equations below).	"yes" or "no". Default is "no".
NASH_SUTCLIFFE	*Optional*. Requests calculation of the Nash-Sutcliffe (1970) coefficient (R^2 in the equations below).	"yes" or "no". Default is "no".
COEFFICIENT_OF_EFFICIENTY	*Optional*. Requests calculation of the coefficient of efficiency (E in the equations below); see Legates and McCabe (1999).	"yes" or "no". Default is "no".
INDEX_OF_AGREEMENT	*Optional*. Requests calculation of the index of agreement (d in the equations below); see Legates and McCabe (1999).	"yes" or "no". Default is "no".
EXPONENT	*Mandatory if either the COEFFICIENT_OF_EFFICIENCY or INDEX_OF_AGREEMENT keyword is present.* The exponent used in the calculation of these quantities (k in th equations below).	An integer – must be 1 or 2.
DATE_1	*Optional*. Terms of the simulated and observed time series before TIME_1 on this date are not used in series comparison.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the simulated and observed time series before this time on DATE_1 are not used in series comparison.	*hh:mm:ss*
DATE_2	*Optional.* Terms of the simulated and observed time series after TIME_2 on this date are not used in series comparison.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of the simulated and observed time series after this time on DATE_2 are not used in series comparison.	*hh:mm:ss*

Keywords within a SERIES_COMPARE block.

```
START SERIES_COMPARE
  CONTEXT all
  SERIES_NAME_SIM mod_flow
  SERIES_NAME_OBS obs_flow
  NEW_C_TABLE_NAME com_series
  BIAS yes
  RELATIVE_BIAS yes
  STANDARD_ERROR yes
  RELATIVE_STANDARD_ERROR yes
  NASH_SUTCLIFFE yes
  COEFFICIENT_OF_EFFICIENCY yes
  INDEX_OF_AGREEMENT yes
  EXPONENT 1
END SERIES_COMPARE
```

A SERIES_COMPARE block.

The names of two time series must be provided in a SERIES_COMPARE block. One of these is denoted as the "observed" time series while the other is the "simulated" time series; the difference is important in calculating relative bias, relative standard error, the Nash-Sutcliffe coefficient, the index of agreement and the coefficient of efficiency, for standardisation of these quantities is undertaken with respect to the observed time series. The simulated and observed time series must contain samples taken at identical dates and times within the time interval spanned by the DATE_1, TIME_1 and DATE_2, TIME_2 entries. If these keywords are not provided the sample dates and times of the observed and simulated time series must be identical over the entire length of these series.

If either of the COEFFICIENT_OF_EFFICIENCY or INDEX_OF_AGREEMENT keywords are present, then an EXPONENT keyword must be present. The theory underpinning use of the coefficient of efficiency and index of agreement as bases for series comparison is discussed in Legates and McCabe (1999). The exponent must be either 1 or 2. If either of these keywords are present, then a SERIES_NAME_BASE keyword can also be supplied, this providing the name of a "baseline time series" that can optionally be used in place of the mean observation value over the comparison time window; see the above reference for details. The baseline time series must have terms at identical dates and times to those of the simulated and observed time series over the comparison time window. If the SERIES_NAME_BASE keyword is omitted, then the mean observation is employed in the formulae presented below instead of the terms of the baseline time series.

Equations for the quantities calculated in the SERIES_COMPARE block are as follows. Note that the Nash-Sutcliffe coefficient is equal to the coefficient of efficiency when the exponent in the latter equation is zero, and when a baseline time series is not provided.

Bias:-

$$B = \frac{1}{N}\Sigma\left(S_i - O_i\right)$$

Standard error:

$$S = \sqrt{\frac{1}{N-1}\Sigma(S_i - O_i)^2}$$

Relative bias:

$$B_r = \frac{B}{\overline{O}}$$

Relative standard error:

$$S_r = \frac{S}{S_o}$$

Nash-Sutcliffe coefficient:

$$R^2 = 1 - \frac{\Sigma(S_i - O_i)^2}{\Sigma(O_i - \overline{O})^2}$$

Coefficient of efficiency:

$$E_k = 1 - \frac{\Sigma|S_i - O_i|^k}{\Sigma|O_i - \overline{O}|^k}$$

Index of agreement:

$$d_k = 1 - \frac{\Sigma|S_i - O_i|^k}{\Sigma(|S_i - \overline{O}| + |O_i - \overline{O}|)^k}$$

where:

$$\overline{O} = \frac{1}{N}\Sigma O_i$$

$$S_o = \sqrt{\frac{1}{N-1}\Sigma(O_i - \overline{O})^2}$$

and N is the number of terms in the series (or subseries) between which comparison takes place; summation in the above equations takes place over all of these terms. Where a SERIES_NAME_BASE keyword is supplied, $\overline{O}$ in the equations for coefficient of efficiency and index of agreement is replace by B_i, the respective term of the baseline time series.

If it is desired that weights be applied to terms of the series before comparison (as is often the case), weighted observation and simulated time series can easily be generated using the SERIES_EQUATION block.

SERIES_DIFFERENCE

The SERIES_DIFFERENCE block is used to calculate a new time series, the terms of which are differences between successive terms of an existing time series. Keywords pertaining to the SERIES_DIFFERENCE block are listed in the table below. An example SERIES_DIFFERENCE block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the SERIES_DIFFERENCE block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series whose terms are to be differenced.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series formed by subtracting subsequent terms of an existing series.	Any character string without internal spaces up to 10 characters in length.

Keywords within a SERIES_DIFFERENCE block.

```
START SERIES_DIFFERENCE
  CONTEXT all
  SERIES_NAME 322
  NEW_SERIES_NAME 322_d
END SERIES_DIFFERENCE
```

A SERIES_DIFFERENCE block.

The outcome of processing a SERIES_DIFFERENCE block is another time series, the terms of which are the differences between successive terms of an existing time series. Where successive terms are subtracted in this manner, the difference is ascribed to the date and time pertaining to the later of the two terms. The new time series therefore has one less term than the original time series, for no term exists in the new time series with the same date and time as that of the first term in the existing time series.

SERIES_DISPLACE

The SERIES_DISPLACE block is used to "migrate" the terms of a series with respect to its time-base, lagging or leading these terms as requested by the user. Keywords pertaining to the SERIES_DISPLACE block are listed in the table below. An example SERIES_DISPLACE block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the SERIES_DISPLACE block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series whose terms are to be displaced.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series formed by term-displacement of an existing series.	Any character string without internal spaces up to 10 characters in length.
LAG_INCREMENT	*Mandatory*. The number of sample intervals by which terms of the original time series are lagged.	An integer.
FILL_VALUE	*Mandatory*. Values assigned to migrated terms at the beginning or end of the time series where no other terms can take their place	A real number.

Keywords within a SERIES_DISPLACE block.

```
START SERIES_DISPLACE
  CONTEXT all
  SERIES_NAME outflow
  NEW_SERIES_NAME outflow_1
  LAG_INCREMENT 1
  FILL_VALUE 0.00
END SERIES_DISPLACE
```

A SERIES_DISPLACE block.

The SERIES_DISPLACE operation is the only procedure undertaken by TSPROC which requires that the time series upon which the operation is carried out have a constant sample interval. If the sample interval is not constant throughout the time spanned by the time series, TSPROC will display an appropriate error message before ceasing execution.

A positive LAG_INCREMENT is used to delay terms in the time series. For example, if a LAG_INCREMENT of 1 is used, then each term within a time series will be assigned to the time and date previously occupied by the term which follows it. If it is desired that terms in the series be shifted in the opposite direction instead, this can be accomplished by using a negative LAG_INCREMENT.

When terms of a time series are shifted in this manner, terms at one end of the series "drop off the edge" (the time-base of the series is not altered by the SERIES_DISPLACE operation). At the other end of the series, at least one term of the shifted series must be assigned a "dummy value" as end positions within the series become vacated by the shifting operation. The user must provide this "dummy value" using the FILL_VALUE keyword.

In undertaking sophisticated (and extremely powerful) parameter estimation procedures such as that described by Kuczera (1983), it is necessary that a combination of an original and a lagged "observed time series" be compared with its model-generated counterpart. Residuals (ie. model-to-measurement differences) achieved through the model calibration process using such combinations of time series are often superior to those achieved using the original time series because the former have drastically reduced inbuilt inter-term correlation structure. The existence of such inter-term correlation can lead to misleading estimates of parameter uncertainty.

A composite series, comprised of an original time series summed with various combinations of lagged time series, can be created using the SERIES_EQUATION block.

SERIES_EQUATION

Through use of the SERIES_EQUATION block a new time series can be formed based on an equation of arbitrary mathematical complexity involving one or a number of other time series. The only two conditions on time series that are cited in this equation are that:-

1. all time series featured in the series equation must have samples at identical dates and times (this can be ensured by using the REDUCE_TIME_SPAN and NEW_TIME_BASE blocks if necessary), and

2. a series equation must feature at least one time series (in order to provide the time-base of the resulting time series).

Keywords appearing in a SERIES_EQUATION block are listed in the following table. An example of a SERIES_EQUATION block follows the table. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the SERIES_EQUATION block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series formed through undertaking the calculations embodied in the series equation.	Any character string without internal spaces up to 10 characters in length.
EQUATION	*Mandatory*. The equation by which the terms of the new series are calculated.	See below.

Keywords within a SERIES_EQUATION block.

```
START SERIES_EQUATION
  CONTEXT all
  NEW_SERIES_NAME new_series
  EQUATION log10(outflow * concentration)
END SERIES_EQUATION
```

A SERIES_EQUATION block.

The terms of the new time series that results from the action of the series equation are computed by implementing the equation on a term-by-term basis on each of the series cited in the equation. Thus each term in the new series is calculated from the corresponding terms of the existing series.

As stated above, the series equation can be of arbitrary complexity, involving any number of terms, and citing any number of existing time series (as long as the above-mentioned time-base-consistency rule is followed). In formulating the equation, the operators "^", "/", "*", "-" and "+" have their usual meanings of "raised to the power of", "division", "multiplication", "subtraction" and "addition"; optionally the "**" operator can be used in place of the "^" operator to signify raising to the power. Operations are carried out in the order indicated above (ie. the normal ordering of mathematical operations); if in doubt, use brackets to set precedence between operators.

An equation supplied in the SERIES_EQUATION block can include most of the commonly-used mathematical functions, viz. *abs*, *acos*, *asin*, *atan*, *cos*, *cosh*, *exp*, *log*, *log10*, *sin*, *sinh*, *sqrt*, *tan* and *tanh*. Note the following:-

1. The *log* function is to base *e*; to calculate logs to base 10, use the *log10* function.

2. The arguments to trigonometric functions must be supplied in radians.

3. Caution must be exercised when using some of these functions that their argument lies within the proper numerical range for that function. For example, if any of the terms of a series upon which a *log* operation is performed are zero or negative, a numerical error will result. TSPROC will detect this error and cease execution with an appropriate error message.

Caution must also be exercised when using the "/" operator that a divide-by-zero condition is not encountered. If this occurs, TSPROC will issue an appropriate error message before ceasing execution.

In addition to the above functions, TSPROC allows two "native TSPROC functions" to be used in a series equation; these are the *@_days_start_year* and *@_days_*"*mm/dd/yyyy_hh:nn:ss"* functions (the "@_" string indicates to the subroutine that parses this equation that the term represents neither a series, a number, nor one of the mathematical functions discussed above).

When the *@_days_start_year* term is encountered in a series equation, the days since the start of the year pertaining to the current series term is substituted for the string. Where a sample does not occur at midnight, fractional days are used in the calculation of the *@_days_start_year* function, the outcome of which is a real number.

When the *@_days_*"*mm/dd/yyyy_hh:nn:ss"* term is encountered, TSPROC calculates the days (as a real number – fractional if necessary) since the indicated date and time. Note that the date and time strings must be collectively enclosed in quotes and must be separated by an underscore. Note also that the correct format to use in expressing the date (ie. *mm/dd/yyyy* or *dd/mm/yyyy*) is determined by the DATE_FORMAT keyword in the SETTINGS block.

The following are some examples of legal series equations.

```
outflow

log10(outflow) + 3.456 * sediment ^ 3.23

34.5 / (interflow + 3.432)

0.0 * series1 + @_days_start_year

3.495 + sin((@_days_start_year + 124.5)*6.284/365.25)

1.0/sqrt(@_days_"1/21/1978_12:00:00")
```

In the third of the above equations the time series named *series1* is multiplied by zero. In this case the series is included in the equation because of the fact that each equation must cite at least one time series in order to set the time-base of the resultant time series. In the fourth of the above equations the argument of the sine function is multiplied by $2\pi/365.25$ in order to achieve periodicity of one year.

Note that, for those not familiar with programming, the equation *a/b*c* is evaluated as (*a/b)*c*. To divide *a* by *b*c* formulate the equation as: *a/(b*c)* or *a/b/c*.

SERIES_STATISTICS

Using the SERIES_STATISTICS block, a number of simple statistics can be calculated from the terms of a time series. Optionally, the terms of the series upon which statistical calculations are based can be limited to those lying within a specified date/time interval. Another option provided by the SERIES_STATISTICS block is for statistics to be calculated on the basis of the log (to base 10) of the terms of the time series, or on the terms of the series raised to an arbitrary power. If it is desired that statistics be calculated on the basis of more complex functions of the terms of a time series, this can be easily achieved by first calculating a new time series using the SERIES_EQUATION block, and then undertaking statistical calculations on the basis of this new time series.

At present, only 8 statistical measures can be calculated using the SERIES_STATISTICS block. These are the mean, standard deviation, sum, maximum, minimum, range the minimum *n* point mean and the maximum *n* point mean. Note that, as is explained below, if it is intended to use any statistics in a calibration exercise undertaken by PEST, then only those statistics that are actually involved in the parameter estimation process should be calculated in a SERIES_STATISTICS block. This, in turn, will limit the output from the LIST_OUTPUT block to only those statistics.

TSPROC stores the outcomes of statistical calculations carried out by the SERIES_STATISTICS block in an s_table. Like other TSPROC entities, each s_table must be provided with a name so that it can be referenced by other TSPROC processing blocks. This name must be 10 characters or less in length and must not include a space character.

Keywords featured in the SERIES_STATISTICS block are listed in the following table. An example of a SERIES_STATISTICS block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is “all”, the SERIES_STATISTICS block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
SERIES_NAME	*Mandatory.* The name of the time series on which statistical calculations will be carried out.	A name of 10 characters or less in length referencing a time series stored within TSPROC’s memory.
NEW_S_TABLE_NAME	*Mandatory.* The name of the new s_table used to store the outcomes of statistical calculations.	Any character string without internal spaces up to 10 characters in length.

SUM	*Optional*. Requests calculation of the sum of the terms of the time series.	"yes" or "no". Default is "no".
MEAN	*Optional*. Requests calculation of the mean of the terms of the time series.	"yes" or "no". Default is "no".
MINMEAN_*n*	*Optional*. Requests calculation of the minimum *n*-count mean of terms of the series.	"yes" or "no". Default is "no".
MAXMEAN_*n*	*Optional*. Requests calculation of the maximum *n*-count mean of terms of the series.	"yes" or "no". Default is "no".
STD_DEV	*Optional*. Requests calculation of the standard deviation of the terms of the time series.	"yes" or "no". Default is "no".
MAXIMUM	*Optional*. Requests calculation of the maximum of the terms of the time series.	"yes" or "no". Default is "no".
MINIMUM	*Optional*. Requests calculation of the minimum of the terms of the time series.	"yes" or "no". Default is "no".
RANGE	*Optional.* Requests calculation of the difference between the maximum and minimum of the terms of the time series.	"yes" or "no". Default is "no".
LOG	*Optional*. Requests that statistics be calculated based on the logs (to base 10) of the terms of the time series.	"yes" or "no". Default is "no". The LOG keyword cannot be used if the POWER keyword is used.
POWER	*Optional*. Requests that statistics be calculated based on the terms of the time series raised to the nominated power.	A real number other than zero. Default is 1. The POWER keyword cannot be used if the LOG keyword is used.
DATE_1	*Optional*. Terms of the time series before TIME_1 on this date are not used in statistics calculations.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_1	*Optional*. Terms of the time series before this time on DATE_1 are not used in statistical calculations.	*hh:mm:ss*
DATE_2	*Optional.* Terms of the time series after TIME_2 on this date are not used in statistical calculations.	*dd/mm/yyyy* or *mm/dd/yy* depending on the DATE_FORMAT setting in the SETTINGS block.
TIME_2	*Optional*. Terms of the time series after this time on DATE_2 are not used in statistical calculations.	*hh:mm:ss*

Keywords in a SERIES_STATISTICS block.

```
START SERIES_STATISTICS
  CONTEXT all
  SERIES_NAME outflow
  NEW_S_TABLE_NAME outflow
  MEAN yes
  STANDARD_DEVIATION yes
  SUM yes
  MAXIMUM yes
  MINIMUM yes
  MINMEAN_5 yes
  MAXMEAN_5 yes
  POWER 0.5
  DATE_1 3/1/1976
  TIME_1 00:00:00
  DATE_2 3/3/1976
  TIME_2 00:00:00
END SERIES_STATISTICS
```

A SERIES_STATISTICS block

Caution should be exercised when using the POWER and LOG keywords. It is illegal for both of these keywords to be present within the same SERIES_STATISTICS block. Furthermore, there is a potential for numerical errors to occur through the use of these keywords. In particular if LOG is set to "yes" and if any of the terms of the time series are zero or negative, TSPROC will cease execution with an appropriate error message. Also if a POWER with an absolute value of less than 1 is supplied and if any of the terms of the time series are negative, or if the POWER is negative and any of the terms of the time series are zero, TSPROC will likewise cease execution with an error message before attempting this impossible calculation.

The MINMEAN_*n* and MAXMEAN_*n* statistics require further explanation. As is apparent from the above example, the user must supply an appropriate value for *n* him/herself. Thus, for example, if MINMEAN_5 is set to "yes", TSPROC calculates the minimum value of the running mean of 5 consecutive values of the series, this calculation taking place over the length of the series, or between the user-provided beginning and end dates. It is important to note that if both the MINMEAN_*n* and MAXMEAN_*n* keywords are supplied in the same SERIES_STATISTICS block, *n* must be the same for both of these keywords. Note also, that the LOG and POWER keywords must not be supplied in the same block as the MINMEAN-_*n* and MAXMEAN_*n* keywords; if this is a problem, use the SERIES_EQUATION block to transform the series prior to use of the SERIES_STATISTICS block.

SETTINGS

The SETTINGS block differs from the other blocks in a TSPROC input file in that it must be the first block listed in this file; furthermore its presence is mandatory.

At the present stage of TSPROC development, only two keywords can be used in a SETTINGS block; both of these are mandatory. See the table below.

Keyword	**Role**	**Specifications**
DATE_FORMAT	*Mandatory*. Determines the format with which dates are represented in TSPROC input and output files.	"dd/mm/yyyy" or "mm/dd/yyyy"
CONTEXT	*Mandatory*. Sets the context for the current TSPROC run, thus determining which blocks in the TSPROC input file are processed.	Any character string without internal spaces of 20 characters or less in length.

Keywords within a CONTEXT block.

```
START SETTINGS
  DATE_FORMAT mm/dd/yyyy
  CONTEXT pest_input
END SETTINGS
```

A CONTEXT block.

The DATE_FORMAT setting allows TSPROC to adapt to the different methods by which the date is represented in different countries. If the month precedes the day, then the date format should be supplied as "mm/dd/yyyy". However if the day precedes the month, then it should be written as "dd/mm/yyyy".

A SETTINGS block can contain only one CONTEXT keyword, the purpose of this being to "set the context" of the entire TSPROC run. Every other block used in a TSPROC input file must contain a minimum of one, and a maximum of five, CONTEXT keywords followed by a character string (of 20 characters or less in length and without internal spaces). If any of these character strings match the CONTEXT character string provided in the SETTINGS block, or if any of these strings is supplied as "all", then that block will be processed.

Use of TSPROC CONTEXT functionality allows the user to vary the tasks carried out by TSPROC by simply varying one entry in its input file, viz. the CONTEXT variable supplied in the SETTINGS block. This can be particularly useful when using TSPROC in conjunction with PEST. In preparing for a PEST run, a user can set up a complex TSPROC input file which processes both measured and model-generated time series, and then generates a PEST

input dataset in which the terms of the processed measured time series act as "calibration targets" to which the terms of the processed model-generated time series are matched. If CONTEXT settings in the various TSPROC processing blocks are carefully selected, it will then be possible for the same TSPROC input file to be used by TSPROC in its capacity as a model post-processor, simply by altering the run CONTEXT in the SETTINGS block.

V_TABLE_TO_SERIES

The V_TABLE_TO_SERIES block copies information stored in a v_table to a new time series. Information stored in time series format has access to more processing functionality than that available for v_tables, including calculation of comparison statistics with other series, digital filtering, time interpolation etc.

Keywords associated with the V_TABLE_TO_SERIES block are listed in the following table. An example of a V_TABLE_TO_SERIES block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the V_TABLE_TO_SERIES block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
NEW_SERIES_NAME	*Mandatory.* The name of the new time series formed through copying entries from a v_table.	Any character string without internal spaces up to 10 characters in length.
V_TABLE_NAME	*Mandatory.* The name of a v_table from which entries are to be copied to the new time series.	A name of 10 characters or less in length referencing a v_table stored within TSPROC's memory.
TIME_ABSCISSA	*Mandatory.* Informs TSPROC whether the date and time corresponding to each new time series entry pertains to the beginning, middle or end of the corresponding v_table interval.	"start", "centre" or "end"

Keywords in a V_TABLE_TO_SERIES block.

```
START V_TABLE_TO_SERIES
  CONTEXT all
  V_TABLE_NAME volume
  NEW_SERIES_NAME ssvol
  TIME_ABSCISSA end
END V_TABLE_TO_SERIES
```

A V_TABLE_TO_SERIES block.

As is apparent when the contents of a v_table are written to file using the LIST_OUTPUT block, there are two dates and times associated with every term of a v_table, ie. the date and time corresponding to the beginning of the interval over which volume was accumulated, and the date and time corresponding to the end of the interval. However there is only one date and time associate with every time series entry. Thus in transferring data between the two entity types, the user must inform TSPROC how time series dates and times are calculated from v_table dates and times. Three options are available:-

1. Time series dates and times can correspond to the beginnings of respective volume accumulation intervals of the v_table from which they are derived;

2. Time series dates and times can correspond to the ends of respective volume accumulation intervals of the v_table from which they are derived;

3. Time series dates and times can correspond to the centres of respective volume accumulation intervals of the v_table from which they are derived.

Selection of the appropriate one of these three options is undertaken by providing the character string “start”, “end” or “centre” (or “center”) with the TIME_ABSCISSA keyword of a V_TABLE_TO_SERIES block.

VOLUME_CALCULATION

The VOLUME_CALCULATION block instructs TSPROC to integrate a time series with respect to time over the time-span bracketed by two dates and times. While the most obvious application of this functionality is in volume calculation, it can also be used for mass calculation if the integration is carried out on a time series which represents the mass flux of some constituent. A mass flux time series can be calculated from time series representing concentration and flow using the SERIES_EQUATION block.

Integration can be carried out over one or multiple time spans. These time spans are defined in a "dates file", the format of which is illustrated below. Dates and times are supplied in a dates file rather than as part of the VOLUME_CALCULATION block because in many instances of model calibration a large number of volumes or constituent masses may be used in the calibration process. In some circumstances integration may take place over regularly spaced (for example monthly) time intervals, whereas in other cases integration may take place over a number of discrete, significant events.

```
03/12/1976  11:23:53    04/03/1976  03:00:00
04/30/1976  12:43:00    09/02/1976  23:59:59
04/30/1976  12:43:00    04/30/1976  23:59:59
```

A dates file.

A dates file can be of any length. Each line must contain 4 entries, viz. the date and time defining the beginning of the integration interval and the date and time defining the end of the interval. The date format must be *dd/mm/yyyy* or *mm/dd/yyyy*; the option chosen must be consistent with the DATE_FORMAT setting in the TSPROC SETTINGS block.

The outcomes of TSPROC's volume calculations are stored in a v_table. Like other TSPROC entities, each v_table must be given a name; this name is supplied through the NEW_V_TABLE_NAME keyword. This, and other keywords associated with a VOLUME_CALCULATION block are listed in the following table. An example of a VOLUME_CALCULATION block follows that. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the VOLUME_CALCULATION block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.

SERIES_NAME	*Mandatory*. The name of the time series on which time integration will be carried out.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
NEW_V_TABLE_NAME	*Mandatory*. The name of a new v_table used to store the outcomes of time-integration carried out by TSPROC.	Any character string without internal spaces up to 10 characters in length.
DATE_FILE	*Mandatory*. The name of the dates file containing the time spans over which time series integration will take place.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
FLOW_TIME_UNITS	*Mandatory*. The time units of flow employed by the time series.	"year", "month", "day", "hour", "min" or "sec"
FACTOR	*Optional*. Factor by which integrated volumes or masses are multiplied before storage	A real number. Default is 1.0.

Keywords in a VOLUME_CALCULATION block.

```
START VOLUME_CALCULATION
  CONTEXT all
  SERIES_NAME outflow
  NEW_V_TABLE_NAME volout
  FLOW_TIME_UNITS days
  DATE_FILE "volume dates.dat"
  FACTOR 3.4953
END VOLUME_CALCULATION
```

A VOLUME_CALCULATION block.

Two VOLUME_CALCULATION keywords require further explanation. The first is the TIME_UNITS keyword; using this keyword, the user must supply the time units employed by the flow time series. For example if flow is recorded in cubic feet per second, then TIME_UNITS should be provided as "sec". The second is the optional FACTOR keyword. With this keyword the user should supply a multiplier which TSPROC applies to each integrated volume or mass which it calculates. The predominant use of this multiplier is in units conversion. For example if it were desired that the volume in cubic feet calculated in the above example be stored in units of acre feet, gallons, megalitres or some other volumetric unit, then the appropriate conversion factor should be supplied.

WRITE_PEST_FILES

General

The WRITE_PEST_FILES block instructs TSPROC to generate PEST input files for a parameter estimation run. Use of this block to generate PEST input files is predicated on the assumption that TSPROC will be used as a model post-processor as part of a composite model (encapsulated in a batch or script file) run by PEST. It is further assumed that the input file supplied to TSPROC when used in this mode is almost identical to that used by TSPROC to generate the PEST input files upon which the parameter estimation process is based. However when used in the latter capacity, a number of items specific to construction of the PEST input dataset are enabled using TSPROC CONTEXT functionality. These processing options must then be disabled once the PEST input dataset has been written, and before TSPROC assumes its role as a model post-processor.

Position within a TSPROC Input File

If present, a WRITE_PEST_FILES block must immediately follow a LIST_OUTPUT block in a TSPROC input file. In writing the PEST input dataset, TSPROC assumes that the LIST_OUTPUT block which immediately precedes the WRITE_PEST_FILES block is exactly the same as that which it will use to generate "model output files" when run as a model post-processor in the forthcoming calibration run. The time series, s_tables, v_tables and e_tables which are cited in the LIST_OUTPUT block are thus classified as the "model" time series, s_tables, v_tables and e_tables. For each of these model-generated entities a corresponding "observation" entity must be supplied. Like the model entities to which they are matched, the observation entities must have been generated (or simply imported) during the current TSPROC run. (Like the WRITE_PEST_FILES block itself, some of the functionality implemented by TSPROC to generate observation entities used by the WRITE_PEST_FILES block will probably be disabled through appropriate CONTEXT selection before the current TSPROC input file is supplied to TSPROC for use in its forthcoming role as a model post-processor.)

Model and Observation Entities

It is important to note that any model entity that is matched to an observation entity must have the same design specifications as that entity. Thus an observation time series must have the same number of terms as the model time series to which it is matched, and each of the terms in these paired time series must pertain to the same date and time. This can be achieved using the REDUCE_TIME_SPAN and NEW_TIME_BASE blocks; using the former block an observation time series can be contracted in length to the time spanned by a model simulation run, while model outputs can be time-interpolated to measurement times using the latter block. Model and observation s_tables must include the same statistics, calculated over the same time spans; however a model s_table will normally have been calculated on the basis of a model-generated time series, whereas an observation s_table will have been calculated on the basis of an observation time series. Similarly, exceedence times contained

in model and observation e-tables must have been calculated from model-generated and observation time series using the same flow thresholds; and model and observation v_tables must have been calculated from model-generated and observation time series using the same set of integration time intervals. Should TSPROC detect any inconsistencies in such paired entities, it will cease execution with an appropriate error message.

It is a very good idea for model s_tables, v_tables and e_tables to be calculated from model-generated time series *after the latter have been time-interpolated to the times and dates of the observation time series to which they correspond*. This is especially important if observations are intermittent and irregular. By doing this, any bias or miscalculation of the quantities stored within the various TSPROC entities is "cancelled out" in the calibration process because both the model and observation quantities are subject to exactly the same error caused by limitations in the time base on which they were calculated. If desired, exact calculations of these quantities can be made on the basis of model-generated time series after the calibration process is complete.

Keywords

The following table describes the keywords associated with a WRITE_PEST_FILES block.

Keyword	**Role**	**Specifications**
CONTEXT	*At least one CONTEXT keyword must be supplied; up to 5 are permitted.* If one of the CONTEXT strings matches the CONTEXT string in the SETTINGS block, or if one of the CONTEXT strings is "all", the WRITE_PEST_FILES block will be processed.	Any string without internal spaces of 20 characters or less in length. The CONTEXT keyword(s) must precede all other keywords.
TEMPLATE_FILE	*Mandatory.* The name of a PEST template file. Use as many TEMPLATE_FILE entries as there are template files involved in the parameter estimation process.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
MODEL_INPUT_FILE	*Optional.* If present, this keyword must immediately follow a TEMPLATE_FILE keyword.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
PARAMETER_DATA_FILE	*Optional*. The name of a file containing data normally found in the "parameter data" section of a PEST control file.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
PARAMETER_GROUP_FILE	*Optional*. The name of a file containing data normally found in the "parameter groups" section of a PEST control file.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.

OBSERVATION_SERIES_NAME	*Optional*. The name of a time series containing measurement data. Must be followed by a MODEL_SERIES_NAME keyword.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
MODEL_SERIES_NAME	*Mandatory for every OBSERVATION_SERIES_NAME keyword.* Model-generated time series corresponding to an observation time series. Must follow an OBSERVATION_SERIES_NAME keyword.	A name of 10 characters or less in length referencing a time series stored within TSPROC's memory.
SERIES_WEIGHTS_EQUATION	*Mandatory for every OBSERVATION_SERIES_NAME keyword.* Equation by which observation weights are calculated. Must follow a MODEL_SERIES_NAME keyword.	An equation, optionally enclosed in quotes.
SERIES_WEIGHTS_MIN_MAX	*Optional*. The minimum and maximum weights for observations pertaining to the previous OBSERVATION_SERIES_NAME keyword. If present, must immediately follow a SERIES_WEIGHTS_EQUATION keyword.	Two real non-negative numbers separated by a space. First the minimum weight, then the maximum weight.
OBSERVATION_S_TABLE_NAME	*Optional*. The name of an s_table containing processed measurement data. Must be followed by a MODEL_S_TABLE_NAME keyword.	A name of 10 characters or less in length referencing an s_table stored within TSPROC's memory.
MODEL_S_TABLE_NAME	*Mandatory for every OBSERVATION_S_TABLE_NAME keyword.* Model-generated s_table corresponding to an observation s_table. Must follow an OBSERVATION_S_TABLE_NAME keyword.	A name of 10 characters or less in length referencing an s_table stored within TSPROC's memory.
S_TABLE_WEIGHTS_EQUATION	*Mandatory for every OBSERVATION_S_TABLE_NAME keyword.* Equation by which observation weights are calculated. Must follow a MODEL_S_TABLE_NAME keyword.	An equation, optionally enclosed in quotes.

S_TABLE_WEIGHTS_MIN_MAX	*Optional*. The minimum and maximum weights for observations pertaining to the previous OBSERVATION_S_TABLE_NAME keyword. If present, must immediately follow an S_TABLE_WEIGHTS_EQUATION keyword.	Two real non-negative numbers separated by a space. First the minimum weight, then the maximum weight.
OBSERVATION_V_TABLE_NAME	*Optional*. The name of a v_table containing processed measurement data. Must be followed by a MODEL_V_TABLE_NAME keyword.	A name of 10 characters or less in length referencing a v_table stored within TSPROC's memory.
MODEL_V_TABLE_NAME	*Mandatory for every OBSERVATION_V_TABLE_NAME keyword.* Model-generated v_table corresponding to an observation v_table. Must follow an OBSERVATION_V_TABLE_NAME keyword.	A name of 10 characters or less in length referencing a v_table stored within TSPROC's memory.
V_TABLE_WEIGHTS_EQUATION	*Mandatory for every OBSERVATION_V_TABLE_NAME keyword.* Equation by which observation weights are calculated. Must follow a MODEL_V_TABLE_NAME keyword.	An equation, optionally enclosed in quotes.
V_TABLE_WEIGHTS_MIN_MAX	*Optional*. The minimum and maximum weights for observations pertaining to the previous OBSERVATION_V_TABLE_NAME keyword. If present, must immediately follow a V_TABLE_WEIGHTS_EQUATION keyword.	Two real non-negative numbers separated by a space. First the minimum weight, then the maximum weight.
OBSERVATION_E_TABLE_NAME	*Optional*. The name of an e_table containing processed measurement data. Must be followed by a MODEL_E_TABLE_NAME keyword.	A name of 10 characters or less in length referencing an e_table stored within TSPROC's memory.
MODEL_E_TABLE_NAME	*Mandatory for every OBSERVATION_E_TABLE_NAME keyword.* Model-generated e_table corresponding to an observation e_table. Must follow an OBSERVATION_E_TABLE_NAME keyword.	A name of 10 characters or less in length referencing an e_table stored within TSPROC's memory.

E_TABLE_WEIGHTS_EQUATION	*Mandatory for every OBSERVATION_E_TABLE_NAME keyword.* Equation by which observation weights are calculated. Must follow a MODEL_E_TABLE_NAME keyword.	An equation, optionally enclosed in quotes.
E_TABLE_WEIGHTS_MIN_MAX	*Optional*. The minimum and maximum weights for observations pertaining to the previous OBSERVATION_E_TABLE_NAME keyword. If present, must immediately follow an E_TABLE_WEIGHTS_EQUATION keyword.	Two real non-negative numbers separated by a space. First the minimum weight, then the maximum weight.
NEW_PEST_CONTROL_FILE	*Mandatory*. The name of the PEST control file to be written by TSPROC.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
AUTOMATIC_USER_INTERVENTION	*Optional.* Determines the "automatic user intervention" setting in the PEST control file written by TSPROC.	"yes" or "no"
TRUNCATED_SVD	*Optional.* Determines whether truncated singular value decomposition will be employed as a stabilization device in the PEST control file written by TSPROC.	A real number greater than zero – the PEST EIGTHRESH variable. Normally about 2.0E-7.
NEW_INSTRUCTION_FILE	*Mandatory*. The name of the instruction file to be written by TSPROC.	Any filename up to 120 characters in length. Use quotes if the filename contains spaces.
MODEL_COMMAND_LINE	*Optional*. The model command line to be recorded in the "model command line" section of the PEST control file.	A command line which satisfies the requirements of the operating system, in the present case the name of a batch or script file.

Keywords in a WRITE_PEST_FILES block.

An example of a WRITE_PEST_FILES block follows. Note that while this example is based on the use of only one time series, s_table, v_table and e_table in the inversion process, others of each of these entities could have been included simply through adding the appropriate sets of keywords and associated entries to the block below.

```
START WRITE_PEST_FILES
  CONTEXT pest_input
  NEW_PEST_CONTROL_FILE case.pst
  AUTOMATIC_USER_INTERVENTION yes
  TEMPLATE_FILE catchment.tpl
  MODEL_INPUT_FILE catchment.uci
  NEW_INSTRUCTION_FILE observation.ins

  OBSERVATION_SERIES_NAME flow_obs
  MODEL_SERIES_NAME i_flow_mod
  SERIES_WEIGHTS_EQUATION 1.0/@_abs_value
  SERIES_WEIGHTS_MIN_MAX 1.0 1000.0

  OBSERVATION_V_TABLE_NAME vol_obs
  MODEL_V_TABLE_NAME vol_mod
  V_TABLE_WEIGHTS_EQUATION 5.0

  OBSERVATION_S_TABLE_NAME stat_obs
  MODEL_S_TABLE_NAME stat_mod
  S_TABLE_WEIGHTS_EQUATION 1.0/@_abs_value

  OBSERVATION_E_TABLE_NAME time_obs
  MODEL_E_TABLE_NAME time_mod
  E_TABLE_WEIGHTS_EQUATION log(2.0/@_abs_value) + 2.0
  E_TABLE_WEIGHTS_MIN_MAX 0 1000

  PARAMETER_DATA_FILE param.dat
  PARAMETER_GROUP_FILE pargroup.dat
  MODEL_COMMAND_LINE model.bat
END WRITE_PEST_FILES
```

A WRITE_PEST_FILES block.

Tasks Undertaken by TSPROC in Generating a PEST Input Dataset

In processing the entries contained within a WRITE_PEST_FILES block, TSPROC undertakes the following tasks.

1. It reads all template files cited in the WRITE_PEST_FILES block, accumulating the names of all parameters cited in those files.

2. If a PARAMETER_DATA_FILE keyword is present within the WRITE_PEST_FILES block, TSPROC reads that file, storing the data contained therein (see below).

3. If a PARAMETER_GROUP_FILE keyword is present within the WRITE_PEST_FILES block, TSPROC reads that file, storing the data contained therein (see below).

4. TSPROC checks that all model entities (ie. time series, s_tables, v_tables and e_tables) cited in the WRITE_PEST_FILES block are also cited in the LIST_OUTPUT block that should immediately precede it in the TSPROC input file.

5. It checks that each observation entity that is matched to a model entity has the same design specifications as its model counterpart.

6. It then generates names for all observations featured in the parameter estimation process (ie. for the individual terms of all time series, and the individual elements of all s_tables, v_tables and e_tables); as is discussed below, these names are based on the names of the respective entities.

7. TSPROC then writes an instruction file by which the "model-generated data" written by the previous LIST_OUTPUT block can be read by PEST.

8. TSPROC then writes the "control data", "parameter group" and "parameter data" sections of the new PEST control file. Included in this file are all parameters referenced in the template files cited in the WRITE_PEST_FILES block. Information contained within the parameter data and parameter group files is included in the pertinent sections of the PEST control file where appropriate. Default values are used for all other PEST variables.

9. The "observation group" and "observation data" sections of the new PEST control file are then written. Observation weights are calculated according to formulae supplied through various WEIGHTS_EQUATION keywords.

10. The "model command line" and "model input/output" sections of the new PEST control file are then written.

These tasks are now discussed in greater detail.

Parameter and Parameter Group Data

TSPROC ascertains the names of the parameters that it must include in the PEST control file by reading all template files cited in the WRITE_PEST_FILES block. Any number of TEMPLATE_FILE keywords can be included in a WRITE_PEST_FILES block. Optionally, each such keyword can be followed by a MODEL_INPUT_FILE keyword. If so, PEST links the cited model input file to the previous template file when writing the "model input/output" section of the PEST control file. If a MODEL_INPUT_FILE keyword is not associated with a particular TEMPLATE_FILE keyword, PEST supplies a default model input filename to correspond to the template file; this filename should be altered to the correct filename in the PEST control file before running PEST.

In writing a PEST control file, TPROC must supply each parameter with an initial value, an upper and lower bound, and all of the other information contained within the "parameter data" section of a PEST control file. It must also assign each parameter to a parameter group. Recall that variables which govern the calculation of derivatives are assigned to parameter groups rather than to individual parameters. For some parameter types, the values assigned to

these derivative-calculation variables can be crucial to the success of the parameter estimation process.

If no PARAMETER_DATA_FILE keyword is present within a WRITE_PEST_FILES block, PEST assigns default values to all parameter variables. It assigns each parameter to a group of its own, and supplies default values to the derivatives-calculation variables pertaining to each such group. The user should carefully inspect all of these variables, altering them as necessary to suite the calibration problem at hand.

If desired, default TSPROC parameter data can be overridden by supplying the values for parameter variables and parameter group variables through a "parameter data file" and a "parameter group file" respectively. The names of these files are supplied following optional keywords of the same name in the WRITE_PEST_FILES block.

A parameter data file is illustrated below.

```
ro1   fixed    factor  0.5   .1  10     ro    1.0   0.0
ro2   log      factor  5.0   .1  10     ro    1.0   0.0
ro3   tied_ro1 factor  0.5   .1  10     ro    1.0   0.0
h1    none     factor  2.0   .05 100    h     1.0   0.0
h2    none     factor  5.0   .05 100    h     1.0   0.0
```

A parameter data file.

For the most part, a parameter data file emulates the "parameter data" section of a PEST control file, containing the same variables in the same order. However, note the following.

1. There is no need to supply a value for the DERCOM variable (the command line number for derivatives calculation - the 10th variable on each line of the "parameter data" section of a PEST control file). TSPROC will always provide a default value of 1 for this variable when it writes a PEST control file.

2. Not all parameters cited in template files need to be cited in a parameter data file. TSPROC will provide default data for parameters that are absent from the latter file.

3. If a parameter is tied to another parameter, the name of the parent parameter must be attached to the "tied" string following an underscore, as illustrated in the above example.

4. If a parameter is assigned to a particular parameter group, and if a parameter group file is not cited in the WRITE_PEST_FILES block, or if the name of the group is not included in a cited parameter group file, then TSPROC will supply default values for variables governing derivatives calculation for that group when it writes the PEST control file.

The contents of a parameter group file emulate those of the "parameter groups" section of a PEST control file. An example is provided below.

```
ro  relative  0.01  0.00    switch 1.5 parabolic
h   relative  0.01  1.0e-4  switch 2.0 parabolic
```

A parameter group file.

Time Series Observations

For every time series involved in the parameter estimation process, at least three, and up to four, keywords must be supplied in the WRITE_PEST_FILES block. These keywords must be provided in the order presented in the above table.

The time series associated with the OBSERVATION_SERIES_NAME keyword should contain measurement data. TSPROC will write the terms of this series to the PEST control file. The goal of the parameter estimation process will be to minimise the discrepancies between these terms and those of a corresponding model-generated time series. The latter will be produced by TSPROC in its role as a model post-processor; as mentioned above, when acting in this latter role CONTEXT settings must be such that a PEST input dataset is NOT generated, and any unnecessary processing of observation data is dispensed with.

The name of the model time series which forms the model-generated counterpart to the observation time series must be supplied with the MODEL_SERIES_NAME keyword directly following the OBSERVATION_SERIES_NAME keyword. It is important to note that this same series must be featured in the LIST_OUTPUT block immediately preceding the WRITE_PEST_FILES block. This LIST_OUTPUT block, and all calculations and data importations giving rise to the time series and tables cited in that block, must be retained when TSPROC is run as a model post-processor during the parameter estimation process.

As was mentioned above, a model time series must have identical specifications to an observation time series with which it is paired, both in the number of terms, and the dates/times pertaining to each of its terms. This will ensure that it is valid to compare the two series on a term-by-term basis for the purposes of calibrating a model.

When writing a PEST input file, TSPROC assigns all observations comprised of the terms of an observation time series to a single observation group. This group is given the same name as the model time series to which the observation time series corresponds. Individual observation names are generated by affixing the string "*#n..n*" to a contraction of the group name, where "*n..n*" is the term number of the time series. If for some reason this process does not result in unique observation names (which can occur under some circumstances if time series names are too similar), TSPROC will inform you of the problem through an appropriate error message and will then cease execution.

When writing the "observation data" section of a PEST control file, TSPROC must assign a weight to each observation. Observation weights are calculated by TSPROC on the basis of

the equation supplied by the user with the WEIGHTS_EQUATION keyword. The format of the weights equation is the same as that described in the SERIES_EQUATION block, except for two important differences. These are as follows.

1. If a series name is cited in a weights equation, that series must have the same time-base (same number of terms, and same date/time pertaining to each term) as the observation time series for which weights are being calculated. In implementing the equation for weights calculation, series are matched on a term-by-term basis.

2. An extra TSPROC-specific function is provided for use in a weights equation that is not available for use in a series equation. This is the *@_abs_val* function. This function returns the value of the term of the observation time series for which a weight is currently being calculated.

Some example weights equations follow.

```
wt_series

1.0/sqrt(@_abs_val)

4.0

1.0 + 0.5 * sin((@_days_start_year + 124.5)*6.284/365.25)

sqrt(@_days_"1/1/1989_00:00:00")
```

In the first of the above equations, weights are simply equated to the terms of an existing time series (which may have been calculated within TSPROC specifically for this purpose). In the second of the above equations, observation weights are calculated as the inverse of the square root of the absolute value of each observation. In the third example a uniform weight of 4.0 is assigned to all observations comprising the observation time series, while in the fourth example weights show a seasonal dependence, being a function of time of year (note the factor of 2π /365.25 in the argument to the sine function). Recall from the documentation to the GET_SERIES_EQUATION block that the argument to the *sin*, *cos* and *tan* functions must be supplied in radians; 2π radians is the same as 360 degrees. In the fifth of the above equations, weights increase as the square root of the number of days that have elapsed since the first moment of 1989.

If any observation weight is calculated as less than zero, TSPROC raises the weight to zero. However the user has the option of supplying upper and lower bounds to the weights him/herself through a SERIES_WEIGHTS_MIN_MAX keyword. (If a user requests a minimum weight of less than 0.0, TSPROC will override this with a minimum weight of zero.)

Note that when generating instructions to read the TSPROC output file whose name is cited in the LIST_OUTPUT block that immediately precedes the WRITE_PEST_FILES block, TSPROC automatically adjusts these instructions according to whether the SERIES_FORMAT is specified as “long” or “short” in that block. Considerable computation time can be saved if the SERIES_FORMAT is “short”.

S_Table Observations

The mechanism by which s_table observations are included in a calibration dataset is very similar to that by which series observations are included in this dataset. The name of an observation s_table must be provided through the OBSERVATION_S_TABLE keyword. This keyword must be immediately followed by a MODEL_S_TABLE keyword through which the name of a corresponding model s_table is provided. This s_table must contain the same statistics as those contained within the observation s_table (statistics for inclusion in an s_table are requested through the SERIES_STATISTICS block). This same s_table must also be featured in the LIST_OUTPUT block immediately preceding the WRITE_PEST_FILES block.

TSPROC assigns all observations pertaining to a particular s_table to a single observation group whose name is the same as that of the model s_table. Individual members of the s_table are provided with observation names by contracting the name of the observation group and appending a shortened form of the name of the statistic which each represents.

Weights for s_table observations are generated using a weights equation. However unlike the weights equation used in the generation of weights for time series observations, the weights equation used for the generation of s_table observation weights cannot site a series name. Nor can it use the *@_days_start_year* or *@_days_*"*mm/dd/yyyy_hh:nn:*ss" functions. However it can use the *@_abs_val* function; in this case the value refers to the particular statistical entity contained in the s_table to which the weight is assigned.

V_Table Observations

V_table observations are included in a calibration dataset in the same way that s_table observations are included. The only difference is that individual observations are named by affixing a number (rather than a contracted form of the name of a statistical measure) to a contracted form of the observation group name. The latter is named after the model v_table to which the observation v_table is matched in the WRITE_PEST_FILES block.

E_Table Observations

Inclusion of e_table observations in the calibration process follows the same procedure as that used for inclusion of v_table observations.

C_Table Observations

As presently programmed, data contained within c_tables cannot be included in the model calibration process. If the name of a c_table is cited in a WRITE_PEST_FILES block, TSPROC will cease execution with an error message.

The PEST Control File

In the PEST control file written by TSPROC, PEST is asked to run in parameter estimation mode. Default values are provided for all PEST control variables. Fortunately, these are

suitable for most occasions; however if a control variable is not suitable for a particular parameter estimation run, it can easily be altered by the user. Similarly, if it is desired that PEST run in another mode, this too can easily be accomplished by direct editing of the PEST control file written by TSPROC. If it is desired that PEST run in regularisation mode, a set of regularisation observations and/or prior information equations must also be added to this file.

If a TEMPLATE_FILE keyword in a WRITE_PEST_FILES block is followed by a MODEL_INPUT_FILE keyword, then the model input file is linked to the corresponding template file in the "model input/output" section of the PEST control file. The name of the instruction file recorded in the "model input/output" section of the PEST control file is that which is written by TSPROC, the name of which is associated with the NEW_INSTRUCTION_FILE keyword. This is matched to the model output file whose name is provided with the FILE keyword in the LIST_OUTPUT block immediately preceding the WRITE_PEST_FILES block.

If a MODEL_COMMAND_LINE keyword is provided in a WRITE_PEST_FILES block, the user-supplied command line is transferred to the "model command line" section of the PEST control file written by TSPROC. Otherwise a default command line is used; this will probably need to be altered by the user before running PEST. Note that the model command line will be the name of a batch or script file. Commands cited in this file will include the name of a model executable, as well as the command to run TSPROC. TSPROC's keyboard responses will need to be written in advance to a small file whose name must be included in the command to run TSPROC following the "<" symbol denoting keyboard re-direction.

The AUTOMATIC_USER_INTERVENTION keyword is used to set the DOAUI variable in the "control data" section of the PEST control file written by TSPROC. If this is set to "yes" then DOAUI is set to "aui"; thus PEST will implement automatic user intervention as necessary when implementing the inversion process. If AUTOMATIC_USER_INTERVENTION is set to "no", or if it is omitted, then DOAUI is set to "noaui". Note that, regardless of the setting of this variable, TSPROC does not add an "automatic user intervention" section to the PEST control file which it writes. Thus if automatic user intervention is implemented, it is done on the basis of default values for variables which control its implementation. If he/she wishes, the user can easily add this section to the PEST control file him/herself.

An alternative numerical stabilisation device is truncated singular value decomposition (i.e. "truncated SVD"). This is invoked by supplying a TRUNCATED_SVD keyword, followed by the value of the truncation threshold (PEST variable EIGTHRESH) which is normally between 10^{-6} and 10^{-7}. Note that TSPROC will object if both a TRUNCATED_SVD and AUTOMATIC_USER_INTERVENTION keyword is supplied, for only one (but not both) of these stabilization devices can be employed at the same time. Note also that if truncated SVD is selected as a numerical stabilization device then the initial lambda (RLAMBDA1 variable) is set to zero and the number of tested lambdas per iteration (NUMLAM variable) is set to 1 in the PEST control file written by TSPROC.

Notwithstanding the fact that it may require some alterations before being used by PEST, a PEST control file written by TSPROC is complete enough to withstand the scrutiny of PESTCHEK. As is described in the PEST manual, PESTCHEK checks both the PEST

control file whose name is provided on its command line, as well as all template and instruction files cited within the PEST control file. Because TSPROC uses parameter names found in one or a number of template files in its construction of the PEST control file, and because it generates the instruction file itself for the current parameter estimation process, PESTCHEK should not detect any errors or inconsistencies in the PEST input dataset built by TSPROC (unless these have been introduced through a spurious parameter data file or parameter group file).

Calibration using "Patterns"

There are some instances of model calibration where the direct matching of raw or processed observation data to corresponding raw or processed model-generated data might not work as well as other strategies for at least some of the data types that may be included in the model calibration process. Certain types of stream quality data fall into this category. For these data types a better calibration strategy may be to attempt to match some relationship between flows and constituent measurements (calculated on the basis of observations on the one hand and model outputs on the other), rather than the individual constituent concentrations themselves. For example the calibration process may attempt to ensure that a regression relationship involving flows, constituent data, and possibly other factors such as time of year, is respected by the model, even if the model is incapable of replicating individual constituent measurements due to the erratic and noisy nature of these measurements.

As an example of the application of this principal, consider that it is "known" that a certain regression relationship exists between flow and constituent concentrations. The coefficients in such a relationship may have been determined through using a model such as the USGS program ESTIMATOR; or they may even have been determined using PEST in conjunction with TSPROC, with the SERIES_EQUATION block of TSPROC comprising the "model". As part of TSPROC's model post-processing duties, model-generated flows and constituent concentrations could be time-interpolated to the dates and times at which constituent measurements were made. Using the SERIES_EQUATION block, the difference between model-calculated concentrations and those "predicted" using the regression equation applied to model-generated flows could be evaluated. The closer that the difference between these two quantities is to zero, the closer does the "constituent pattern" generated by the model match the observed "constituent pattern". (Other factors will come into play here, such as the average and standard deviation of the constituent measurements which, as discussed above, are also easily incorporated into the parameter estimation process.)

In order to incorporate "pattern matching" of this type into the parameter estimation process, a time series expressing the difference between modelled constituent concentrations and those calculated from modelled flows using the "known" regression equation can be supplied as a model time series in the WRITE_PEST_FILES block (and the LIST_OUTPUT block preceding it). For consistency, dates and times for this time series should correspond only to constituent measurement times. The corresponding observation time series would be one with an identical time-base, but with all terms equal to zero. Weights assigned to these "observations" could be uniform; alternatively they could be a function of the actual observed constituent concentrations, calculated using a SERIES_EQUATION block and supplied through the SERIES_WEIGHTS_EQUATION keyword.

References

Baier, G., Cohn, T. and Gilroy, E., 1995. Instructions for using the ESTIMATOR software. Downloaded from the "Unofficial Estimator2000 Home Page" at:- http://www159.pair.com/cohns/TimCohn/TAC_Software/Estimator

Kuczera, G., 1983. Improved parameter inference in catchment models. 1. Evaluating parameter uncertainty. Water Resources Research, v 19, no. 5, pp 1151-1172.

Legates, D.R. and McCabe, G.J., 1999. Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. Water Resources Research, v35, no. 1, pp233-241.

Nathan, R.J. and McMahon, T.A., 1990. Evaluation of automated techniques for base flow and recession analysis. Water Resources Research, v 26, no. 7, pp1465-1473.

Nash, J.E. and Sutcliffe, J.V., 1970. River flow forecasting through conceptual models. Part 1: A discussion of principles. J. Hydrol. 10, 282-290.

Appendix A

Documentation of Other Utility Programs

ADJOBS

Function of ADJOBS

ADJOBS is an acronym for "adjust observations". ADJOBS reads an existing PEST control file. It allows the user to perform the following tasks:-

- introduce new observation groups on the basis of observation names, and
- calculate observation weights on the basis of observation values; different formulae can be used for weights calculation for different observation groups.

ADJOBS is useful in preparing for a PEST run in conjunction with a model that produces one or a number of lengthy time series. In this capacity it is complementary to programs TSPROC and PESTPRP1.

Using ADJOBS

ADJOBS commences execution by prompting the user for the name of an existing PEST control file. Before being read by ADJOBS, this file should have been checked using PESTCHEK for, while ADJOBS will detect and report many of the types of errors that may be present in a PEST control file, its checking functionality is not as complete as that of PESTCHEK.

ADJOBS then reads the PEST control file whose name has been supplied. It pays particular attention to the "observation groups" and "observation data" sections of this file. It counts the number of observations belonging to each observation group and then asks the user a series of questions pertaining to each such group.

First:-

```
Observation group "obsgp1" (245 observations belong to this group) ---->
  Do you wish to make any adjustments  [y/n]?:
```

If you do not wish to subdivide this group into other observation groups, or to re-assign weights to the members of this group, answer "n" to the above prompt. However if you answer "y", ADJOBS will then prompt:-

```
Divide group into subgroups  [y/n]?
```

Division into subgroups takes place on the basis of observation names. Thus if different observation types have different name types, these different observation types can be easily placed into different groups. For example if discharge observations are named *dis001, dis002, dis003,* etc. and pollutant concentrations are labelled *conc001, conc002, conc003,* etc, and both of these observation types presently belong to the same group, they can be separated on the basis of the first three letters of each of their observation names, with observations beginning with the letters "*dis*" being assigned to one group and those beginning with the

letters "*con*" being assigned to another group. To achieve this, answer the following prompt with the number "3":-

```
Use first n characters of observation name for group definition.
Enter n:
```

ADJOBS inspects the names of all of the observations belonging to the current group. It ascertains the number of groups into which the present group is now subdivided and, for each such group, it asks:-

```
Observations in group "obsgp1" beginning with "dis" --->
Enter observation group name for these observations:
```

Provide an observation group name as requested. Note that the user may, in response to this or any other prompt, press the "e" key followed by return. This provides an "escape" mechanism, returning the user to the previous prompt.

After a new observation group name has been supplied, ADJOBS prompts for the variables required for weights calculation. If the user desires, weights can be calculated differently in each new group. ADJOBS prompts:-

```
Adjust weights for this observation subgroup  [y/n]?
```

Then, if "y",

```
Weights are calculated as w = a*(abs[observation_value])**b+c
 Enter a:
 Enter b:
 Enter c:
 Enter maximum allowable weight:
 Enter minimum allowable weight:
```

Respond to these prompts as appropriate; note that the formula for weights calculation is:-

$w = a[abs(observation\_value)]^b + c$

By supplying *b* as –1, weights can be made inversely proportional to observation values. This is often useful when estimating parameters for a runoff-generation model on the basis of observed values of discharge. All calculated weight values are constrained to lie within the maximum and minimum bounds provided by the user; neither of which can be less than zero.

Note that weights can be adjusted in this manner whether or not a group is subdivided into smaller groups. The user simply informs ADJOBS that he/she does not wish to carry out group subdivision; he/she will then be prompted for the parameters of the weights calculation formula for the entire group.

Finally, ADJOBS prompts for the name of a new PEST control file. It then writes this file using the new observation group names and/or weights provided by the user. You should check this file using PESTCHEK before supplying it to PEST.

Uses of ADJOBS

As has already been mentioned, ADJOBS is particularly useful in PEST pre-processing where PEST is to be used in the estimation of parameters for a model which generates one or a number of lengthy time series, particularly flow time series. Because there are often many

measurements comprising an observation dataset in this context, preparation for a PEST run requires software assistance. Such assistance is available through TSPROC and PESTPRP1, whose use is documented elsewhere in this manual.

In many instances of parameter estimation on the basis of discharge time series, it is wise to vary weights in accordance with discharge values. It is often useful to set weights in inverse proportion to discharge in order that the objective function is not dominated by high flows. This can be achieved through supplying a negative exponent in the weight generation equation. Note that if this is done, it is wise to set appropriate upper and lower bounds on weights by responding to ADJOBS prompts in the appropriate manner.

See Also

See also PESTPRP1, SMP2SMP and TSPROC.

IQQM2SMP

Function of IQQM2SMP

IQQM2SMP reads unformatted files produced by IQQM (Integrated Quality and Quantity model produced by the New South Wales Department of Land and Water Conservation). It converts data of user-nominated types at user-nominated sites to site sample file format. Thus if IQQM2SMP is run after IQQM as part of a composite model, selected IQQM output datasets are available in site sample file format. As such, they are amenable to plotting using SMP2HYD and volumetric calculation using SMP2VOL. Most importantly, however, they are also amenable to processing using TSPROC, this greatly facilitating the use of IQQM in conjunction with PEST.

Using IQQM2SMP

Because IQQM2SMP uses dates and times, it is essential that a settings file named *settings.fig* reside in the subdirectory from which IQQM2SMP is run. Recall that the contents of this file inform any utility which reads it of the format to use when reading and writing dates, in particular whether the day precedes the month or vice versa. If a settings file is not present in the subdirectory from which IQQM2SMP is run, IQQM2SMP will terminate execution with an appropriate error message.

Use of IQQM2SMP requires that the user be aware of the contents of IQQM unformatted direct access output files. For version 6.31 of IQQM, direct access output files are named **n.out* where *n* is [00] to [12], this number representing the node type; there is thus a different output file written for each node type. Within each such file, data is stored for each of a number of data types for each pertinent node. (Data types for storage are selected using the IQQM "output" menu.) Thus at each IQQM output time, *m* variables are written to each IQQM direct access output file, where *m* is the number of nodes of the pertinent type for that file multiplied by the number of data types for which output was requested for that node type. IQQM output can be generated at either hourly or daily intervals.

IQQM2SMP begins execution by prompting for the name of the IQQM unformatted output file which it must read:-

```
Enter name of IQQM output file:
```

It then prompts for the starting and finishing dates of the simulation. IQQM2SMP must be supplied with exactly the same starting and finishing dates as were supplied to IQQM when it commenced the model run which generated the files. The next prompt is:-

```
Enter number of output times per day:
```

The response to this prompt depends on the value supplied for *Odt* on the IQQM input file. If *Odt* is set to 1 hour, there will be 24 output times per day; if it is set at 24 hours, there will be only 1 output time per day.

IQQM2SMP's next prompt is:-

```
Pre-read IQQM file to determine number of variables in file? [y/n]:
```

If the user wishes, IQQM2SMP will quickly read the IQQM direct access output file to determine how many records are in this file. It will then divide the number of records by the number of simulated days times the number of outputs per simulated day to evaluate the number of output variables stored in this file. This will be equal to $m \times d$ where m is the number of nodes of the pertinent type cited in the output file and d is the number of data types recorded for each such node. If the user responds to the above prompt with "y", IQQM will write a number of lines to the screen similar to the following.

```
Details of file ine500.out:-
   Number of days of stored data  = 1474
   Number of records              = 165088
   Number of data output times    = 1474
   Number of variables            = 112
```

The number of days of stored data is calculated according to the simulation beginning and end dates as supplied previously to IQQM2SMP by the user. The item of most interest in the above table from the point of view of answering the ensuing prompts is the number of variables. If there is any inconsistency between the user-supplied simulation dates, the user-supplied number of output times per day, and the number of records found in the file, such that the number of variables cannot be calculated as an integer in the manner described above, IQQM2SMP will report an appropriate error message and cease execution.

If you already know the number of variables residing in the IQQM-generated unformatted file, then this section of IQQM2SMP execution can be by-passed by answering "n" to the above prompt. This will normally be the case when using IQQM2SMP as part of a composite model run by PEST. Informing IQQM2SMP of the total number of variables (thus eliminating the need for it to determine this itself) saves IQQM2SMP the trouble of having to read the IQQM output file twice each time it is run. So if the user answers "n" to the above prompt, IQQM2SMP prompts:-

```
Enter the number of variables stored in file filename:
```

where *filename* is the name of the IQQM unformatted, direct access output file.

IQQM2SMP next asks:-

```
For how many variables do you wish to extract data:
```

Supply an integer equal to the number of variables that you would like represented in the site sample file generated by IQQM2SMP. It this is supplied as p, then IQQM2SMP issues the following set of prompts p times:-

```
Variable number 1 for IQQM2SMP output file:-
   Enter IQQM variable number:
   Enter identifier for variable [10 characters or less]:
```

The variable number can be calculated from the data output sequence represented in the IQQM output file. Thus if the 5th data type for the 4th node cited in the file is required, and 12 data types are written to the output file for each of these kinds of node, then the variable

number will be 41 ie. (4-1) × 12 + 5. The variable number must not exceed the number of variables recorded in the file as either calculated by IQQM2SMP or as supplied by the user.

The identifier to be supplied in response to the second of the above prompts is the "site identifier" that will be recorded for that data type on the site sample file generated by IQQM2SMP. In accordance with site sample file protocol, supply a name of 10 characters or less in length.

Next IQQM2SMP prompts for the name of the site sample file which it must generate. Once this is supplied, IQQM2SMP writes this file and terminates execution.

As pointed out in the first section of this manual, at any stage of execution the user can return to the previous prompt by responding to the current prompt with "e" (for escape) followed by the <Enter> key.

If there is any doubt in the user's mind surrounding the issue of whether the correct variable number was supplied in order to extract data of the required type for the required node, the matter can soon be resolved by comparing the site sample file produced by IQQM2SMP with a text file containing the same data produced by selection of the "List" option of the IQQM menu. The text file produced by this latter method will contain fewer significant figures than data written by IQQM2SMP; however a comparison of the two files will quickly reveal whether the data contained in each is the same.

An inspection of a site sample file produced by IQQM2SMP may reveal something that at first appears like an inconsistency; however a closer inspection will reveal that nothing is amiss. Dates recorded in the IQQM2SMP output file may appear to be delayed by one day from those used by IQQM. However it should be noted that site sample file format requires that midnight at the end of one particular day is represented as zero hours on the following day. Thus if the user is at first confused by the fact that there appears to be a date represented in the IQQM2SMP output file that actually postdates the end of IQQM simulation by a day, a closer inspection will reveal that the time associated with this date is 00:00.00, this being midnight on the last day of the simulation.

Uses of IQQM2SMP

Once IQQM-generated data is in site sample file format, it is available for further processing using other members of the PEST Surface Water Utilities documented in this manual. For example SMP2HYD can be used to re-caste IQQM output data into a form where it is ready for plotting using commercial scientific plotting software. Volumetric calculations can be carried out using SMP2VOL or TSPROC.

If IQQM calibration is being undertaken using PEST, then use of IQQM2SMP becomes indispensable. Once IQQM outputs have been converted to site sample file format, TSPROC can be used as a model post-processor and as a PEST input file generator for calibration exercises of arbitrary complexity.

See Also

See also PESTPRP1, SMP2HYD, SMP2SMP, SMP2VOL, and TSPROC.

PD_MS1

Function of PD_MS1

PD_MS1 is a PEST "driver"; that is, it runs PEST many times. On each occasion that it runs PEST, it selects a different set of initial parameter values. These are selected randomly using a uniform or log-uniform distribution over a user-selectable range. After each PEST run it reads from the PEST run record file the optimised objective function, the contribution made to the optimised objective function by each observation group, as well as the maximised/minimised model prediction (if PEST is running in predictive analysis mode). These are recorded on the PD_MS1 run record file, together with initial and optimised parameter values for that run (the latter having been read from the PEST parameter value file).

Using PD_MS1

PD_MS1 commences execution with the prompt:-

```
Enter name of existing PEST control file:
```

Enter the name of a PEST control file, after checking its integrity with PESTCHEK. This file can be of arbitrary complexity, and can instruct PEST to run in either parameter estimation or predictive analysis mode. However if it instructs PEST to run in regularisation mode, PD_MS1 will terminate execution with an appropriate error message.

PD_MS1 reads the "parameter data" and "observation group" sections of the nominated PEST control file. Its task is to initialise the PEST optimisation process from many different, randomly-selected points in "adjustable parameter space". This is defined as the space occupied by parameters which are neither fixed nor tied. PD_MS1 will always supply the same initial value to a fixed parameter, this being the value supplied in the PEST control file. Tied parameters are linked to the randomly-selected values of their parent parameters using the same ratios as provided in the existing PEST control file; hence correct parameter relativity is maintained.

PD_MS1 next asks the user:-

```
Enter number of points for parameter random sampling:
```

This is equivalent to the number of PEST runs which will be carried out under the supervision of PD_MS1. For each such run, PD_MS1 generates a random sample of adjustable parameter space as the starting point for the optimisation process. For each adjustable parameter it prompts:-

```
Parameter lower/upper bounds for pre-inversion sampling must be supplied:-
   bounds for "par1" (Press <Enter> if [.100000, 10000.0]):
   uniform or log sampling over this interval? [u/l] (<Enter> if uniform):
```

("par1" in the second of the above lines is replaced by the pertinent parameter name.) Default sampling bounds supplied for each parameter in the above prompts are read from the

PEST control file. The user can accept these bounds by simply pressing the <Enter> key, or he/she can override them by providing sampling bounds of his/her own; note, however, that PD_MS1 insists that user-supplied lower and upper sampling bounds do not lie beneath or above, respectively, parameter bounds listed in the PEST control file. Random sampling from between these sampling bounds is undertaken using a uniform probability distribution applied either to the native parameters (default) or the logs of these parameters. (Note that the logarithm option is not available for parameters which have negative lower bounds.)

After it has issued the above prompts for every adjustable parameter (that is, for every parameter that is neither tied nor fixed), PD_MS1 prompts for the name of a "trial PEST control file":-

```
Enter name for trial PEST control file (<Enter> if t###.pst):
```

This file is a direct copy of the existing PEST control file, except for the fact that initial parameter values are randomly selected in the manner described above. Press <Enter> to accept the default.

In order to run PEST, PD_MS1 needs to know the pertinent command. This will often be simply "pest". (PD_MS1 will add the name of the PEST control file to the PEST command itself.) However if the name of the directory in which the PEST executable is situated is not cited in the system PATH, the command "pest" will not work. Hence the user is given the opportunity of providing PD_MS1 with a command that includes this path. The prompt is:-

```
Enter command to run PEST (<Enter> if "pest"):
```

Next PD_MS1 prompts for the name of its run record file:-

```
Enter name for PD_MS1 run record file (<Enter> if pd_ms1.rec):
```

for which *pd_ms1.rec* is the default.

PD_MS1 next runs PEST. You should see PEST's screen output scrolling past as it runs. At any time, the outcomes of PEST runs carried out up to that moment are available through inspection of the PD_MS1 run record file.

For each run which it undertakes, PD_MS1 records on its run record file parameter starting values (which are randomly selected as discussed above), optimised parameter values, the optimised objective function, the contribution made to the optimised objective function by different observation groups, and (if PEST is running in predictive analysis mode), the maximised/minimised prediction. Note that the contribution made to the objective function by any prior information that is not assigned to an observation group is not recorded on the PD_MS1 run record file; *hence all prior information should be assigned to an observation group prior to running PD_MS1*.

Uses of PD_MS1

PD_MS1 is useful for examining whether the outcomes of a parameter estimation or predictive analysis process are dependent on initial parameter values. The existence of local objective function minima may give rise to this phenomenon. If that is the case, PEST's

ability to find the global optimum quickly and efficiently will be enhanced through the use of another PEST driver, PD_MS2.

See Also

See also PD_MS2.

PD_MS2

Function of PD_MS2

PD_MS2 is a PEST driver; that is, it runs PEST many times. ("PD_MS" stand for PEST Driver – Multiple Search.) Its task is to seek the global minimum in the objective function in a calibration context in which there may be many local minima. It accomplishes this task through a combination of random parameter sampling, running PEST from different starting locations in parameter space, and "learning from previous mistakes". Not only can PD_MS2 find the global objective function minimum relatively quickly in many calibration contexts compared to other parameter estimation methods; it can also be used to rapidly probe parameter space in order to establish the locations of these other minima.

Using PD_MS2

Conceptual Underpinnings of PD_MS2

The strength of global optimisation methods such as SCE-UA, genetic algorithms, simulated annealing and other such methods, is their ability to incorporate a certain degree of randomness into the search for the minimum of the objective function. There are costs and benefits associated with this strategy. The computational cost of adding such randomness to the selection of trial parameter values lies in the fact that many such values must often be selected for these methods to work properly; the selection of every such parameter set requires that a model run be undertaken in order to evaluate the objective function associated with that set. However the benefit of this strategy lies in the fact that it is only through the inclusion of such randomness in the global search strategy that it can be guaranteed that all "regions of attraction" within parameter space are "felt" at various stages of the optimisation process; the global objective function minimum lies at the base of one of these regions of attraction. The efficiency with which different global optimisation methods can locate this global minimum depends in part on the extent to which they can "learn from experience" in pursuing parameter trajectories which lead towards local, rather than global, objective function minima.

The Gauss-Marquardt-Levenberg (GML) method employed by PEST is not a global optimisation method. Where there is only one minimum of the objective function, the GML method is normally the fastest way to find it, involving far fewer model runs than any other method. However if parameter starting values are located in the catchment area of a local minimum, rather than the global minimum, then the GML method will very efficiency find its way to the local minimum.

PD_MS2 attempts to combine the strengths of global optimisation methods with the speed of the GML method by incorporating a certain degree of randomness into the parameter estimation process, together with an ability to "learn from previous mistakes". The result is a method that, in many incidences, is much more efficient than traditional global methods, while still retaining the robustness that characterises such methods.

PD_MS2 commences the global optimisation process by selecting points in parameter space at random, and then carrying out model runs on the basis of these points. (It does this by running PEST with NOPTMAX – the number of optimisation iterations – set to zero.) Normally a number of points between 4 and 10 times the number of adjustable parameters is warranted in this random selection and testing process. For each random parameter set, PD_MS2 records the resulting objective function. Parameter sets are then ordered according to increasing objective function.

Next PD_MS2 undertakes a series of PEST optimisation runs using some of these randomly-generated parameter sets as starting values. For its first optimisation run it selects the parameter set corresponding to the lowest objective function. After this initial optimisation run is complete, PD_MS2 reads the optimised objective function, together with optimised parameter values from pertinent PEST output files. It also reads parameter values calculated by PEST at each iteration of the GML parameter estimation process which it implements. Thus it is able to record the "parameter trajectory", that is the path taken by parameters as they are upgraded from their initial values to those which minimise the objective function. It can be presumed that if starting values are selected for another PEST run which are anywhere near this trajectory, PEST will find the same objective function minimum. Thus whether or not this is a local minimum (which will be unknown until other minima are found, should they exist), PD_MS2 must select an initial parameter set for the next PEST run which is as far as possible from this previous parameter trajectory if it is to minimise the probability of finding the same minimum. However because experience has demonstrated that "nuisance" regions of attraction can often exist in far-flung parts of parameter space where many parameters are near their bounds, some caution must be exercised in maximising distance in parameter space from this previous trajectory. Hence PD_MS2 restricts its search for another set of starting parameter values to those members of the initial collection of randomly-sampled parameters whose corresponding objective functions are lower than the median for the entire collection of random parameter samples.

Thus for each randomly-selected parameter set from the original sample for which the objective function is better than the median, PD_MS2 calculates the distance of closest approach of that set to the previous parameter trajectory. This "distance" is calculated as the Euclidean distance (that is, the square of the sum of the squared component differences) in parameter space, with each component difference normalised by dividing by the difference between the user-supplied upper and lower sampling bounds for that parameter. The user is also given the choice of whether inter-parameter distances are to be measured using native or log-transformed values for each parameter. The parameter set is then selected for which the distance of closest approach is a maximum. That parameter set is then used to initialise another PEST run.

When the next PEST run is complete, the process is repeated; however the next parameter starting point is selected to be maximally distant from ALL previous parameter trajectories, thus minimising the probability that the parameter starting values for the next PEST run are within the catchment area of any previously defined local minimum.

PD_MS2 ceases to carry out further PEST runs when it appears unlikely that such runs will find a parameter set which leads to the calculation of an objective function that is lower than

any that have already been found. The termination criteria which govern this decision-making process are set by the user.

It should be noted that in selecting starting parameters for PEST runs, PD_MS2 respects the fact that certain parameters may be tied or fixed. In the case of tied parameters, user-supplied parameter ratios are maintained; fixed parameters are not subject to random selection, and are not varied from their assigned values in the parameter estimation process.

Running PD_MS2

PD_MS2 commences execution with the prompt:-

```
Commence a new PD_MS2 run or restart an old one?  [n/o]:
```

PD_MS2's restart functionality will be discussed shortly. The usual response to the above prompt is "n". Next PD_MS2 prompts:-

```
Enter name of existing PEST control file:
```

PD_MS2 reads parameter values, bounds and their fixed/tied status from this file. Then it prompts:

```
Enter number of points for pre-inversion random sampling:
```

This number must be at least twice the maximum number of PEST runs which PD_MS2 is allowed to undertake (the value for this variable is supplied by the user shortly). As mentioned above, it is suggested that it also be at least four times the number of adjustable parameters (ie. parameters which are neither tied nor fixed) cited in the existing PEST control file.

For each adjustable parameter, PD_MS2 issues the following three prompts:-

```
Parameter lower/upper bounds for pre-inversion sampling must be supplied:-
    bounds for "par1" (Press <Enter> if [.100000, 10000.0]):
    uniform or log sampling over this interval? [u/l] (<Enter> if uniform):
    measure distances in natural or log space? [n/l] (<Enter> if natural):

    etc
```

For each one of these prompts default values are supplied; simply press the <Enter> key to accept the default in each case. (Press the "e" key and then <Enter> to backtrack, in accordance with normal practice when using the PEST Surface Water Utilities.)

Random selection of parameter values for potential initialisation of PEST runs takes place according to a uniform or log-uniform distribution, with default upper and lower sample bounds being equal to the upper and lower parameter bounds cited in the PEST control file; the user can alter these sampling bounds, but is not allowed to supply sampling bounds which transgress the parameter bounds cited in the original PEST control file. Note also that the log-uniform random selection option is not available for parameters whose lower bound is negative. The last of the above three prompts pertains to the way in which the distances between sample points and previous parameter trajectories are measured for each parameter. As indicated, this distance can be measured in log space or natural space. In either case it is normalised by dividing by the distance (measured in log or natural space) between parameter sampling bounds.

Once the above information has been supplied for all adjustable parameters, PD_MS2 prompts for the name of the "trial PEST control file" which it will use to run PEST; this is a direct copy of the existing PEST control file except for the fact that initial parameter values are altered in accordance with PD_MS2's current selection for these values. Also, when undertaking initial model runs on the basis of randomly-selected parameter sets for the sole purpose of objective function calculation, the PEST NOPTMAX variable (number of model runs) is set to zero in the trial PEST control file. The prompt is:-

```
Enter name for trial PEST control file (<Enter> if t###.pst):
```

Next PD_MS2 prompts for its termination criteria:-

```
Enter maximum PEST inversion runs to carry out  [<Enter> if 6]:
Enter maximum inversion runs with no improvement  [<Enter> if 3]:
Enter phi improvement fraction judged to be negligible  [<Enter> if .05]:
```

If the above default criteria are selected, PD_MS2 will undertake a maximum of 6 PEST runs. However if 3 runs have elapsed without an objective function improvement of at least 5%, PD_MS2 will terminate execution in the belief that the global objective function minimum has been found.

Next PD_MS2 prompts:-

```
Enter command to run PEST (<Enter> if "pest"):
```

If the PEST executable resides in a directory which is cited in the PATH environment variable, the default command "pest" will be satisfactory; if it does not, the full path to the PEST executable may need to precede its name when responding to the above prompt.

Next PD_MS2 prompts for the name of its run record file, for which the default is *pd_ms2.rec*; finally, it prompts for the name of its restart file, for which the default is *pd_ms2.rst*. Once a response to this prompt has been provided PD_MS2 commences execution, first using PEST to undertake single model runs on the basis of randomly sampled parameter sets, and next to perform objective function minimisation. The PD_MS2 run record file records random parameter samples and objective functions calculated during the initial phase of PD_MS2 execution (when PEST is used to undertake single model runs), as well as optimised parameter values and optimised objective functions calculated by PEST during its later execution phase (in which PEST is run in order to calculate optimal parameter values). Parameters corresponding to the global objective function minimum are recorded at the end of the PD_MS2 run record file.

A Note on Efficiency

As discussed above, before using PEST to minimise the objective function, PD_MS2 calculates objective functions corresponding to a series or randomly selected parameters. It does this by running PEST with NOPTMAX set to zero; thus PEST takes care of writing parameters to model input files, reading numbers from model output files, and of calculating the objective function from model-to-measurement residuals. Where model run times are short this can be a very inefficient procedure, because in order to undertake a model run PEST must first be loaded into the computer's memory, and all of its input files must be read. Where many observations are involved in the parameter estimation process, this may take a

moment. As this same task is then repeated for every model run, the overheads involved in making each such run can slow the process down. Where model run times are significant, repeated re-loading of PEST will not be an impost. However where model run times are small, the PEST re-loading time can be significant in comparison with the model run time. Future versions of PD_MS2 will overcome this difficulty by allowing it to communicate directly with a model, and to calculate the objective function itself without the aid of PEST. In the meantime, please contact me if it causes a problem, and I will update PD_MS2 sooner than I otherwise would.

Running PEST in "Regularisation" Mode

PD_MS2 can be used in conjunction with PEST running in regularisation mode. In this case, PD_MS2 monitors the measurement objective function rather than the total objective function. Care should be taken when running PEST in regularisation mode in conjunction with PD_MS2 however. Reasons for this are:-

1. If regularisation is properly implemented, the number of regions of attraction in parameter space should, hopefully, be reduced to just one.

2. When running in regularisation mode, the user sets a target objective function (PEST variable PHIMLIM) below which PEST will not reduce the measurement objective function. There may indeed be different ways to achieve this target objective function, with points in parameter space through which this objective function is achieved forming a valley rather than a set of discrete minima. However, numerical noise in the regularised inversion process may result in some scatter about PHIMLIM, and thus present the illusion of local minima, when in fact the locus of points for which the objective function is equal to PHIMLIM is continuous.

Restarting a PD_MS2 Run

As it runs, PD_MS2 stores the values of many of its variables in a binary "restart file", the default name of which is *pd_ms2.rst*. If PD_MS2 execution is interrupted in mid run it can then be re-started without the necessity of repeating all calculations undertaken up to the point of interruption.

When re-commencing an interrupted PD_MS2 run, answer "o" to the first of its prompts (to signify re-commencement of an old run). PD_MS2 will then prompt for the name of the "restart file" written on its previous run. Provide this name (or simply press <Enter> to accept the default).

PD_MS2 then issues the prompt:-

```
Run PEST with "/r", "/s", "/j" or no switch on first run: [r,s,j,n]:
```

PD_MS2 can be interrupted in two ways. The "brutal" way is to press the <Ctl-C> keys. The more civilised approach is to open another command-line window in the PD_MS2 working directory and type "PSTOP". This will interrupt PEST execution (after the current model run has finished). PD_MS2 will cease execution shortly thereafter with an error message to the effect that it cannot find a completed PEST control file. When PD_MS2 is re-started, the

interrupted PEST run can then be restarted if, on the first PEST run of renewed execution, PD_MS2 runs PEST with a restart switch ("/r" or "/j", or possibly "/s" if PD_MS2 is running the parallel version of PEST). If PEST run times are long, this can save a considerable amount of time, for PEST will not have to start the run on which its execution was interrupted from the beginning; rather it can re-start this run from shortly before the point of its interruption.

It is important to note that no restart information is saved by PD_MS2 until the end of the first PEST parameter estimation run. Thus if PD_MS2 is interrupted during the pre-inversion random parameter selection stage, or during the first PEST inversion run, no binary file will be available for a mid-run restart.

It is the user's responsibility not to alter the PEST control file upon which PD_MS2's execution is based between halting a PD_MS2 run and re-starting it.

Uses of PD_MS2

PD_MS2 has the potential to be of great use in parameter estimation contexts which are prone to the occurrence of local objective function minima. Such minima are commonly encountered in the calibration of surface water models. The use of PD_MS2, rather than simply PEST, in the calibration of such models, allows the user to gain confidence that an objective function minimum is indeed global. Alternatively, it allows the user to rapidly find multiple minima within parameter space (or perhaps different locations along a "valley" in parameter space). Knowledge of the existence and whereabouts of such minima may be invaluable in assessing the outcomes of a parameter estimation process.

See Also

See also PD_MS1, SCEUA_P.

PESTPRP1

Function of PESTPRP1

Program PESTPRP1 undertakes the laborious task of preparing a PEST control file and a PEST instruction file prior to using PEST with a model which generates an output time series. The laborious nature of this work arises from the fact that a great deal of observation data normally needs to be processed; however with PESTPRP1 the entire process is automated. It is assumed that the "model" to be calibrated consists of the simulator followed by a program that translates model output data to site sample file format (eg. PLT2SMP or IQQM2SMP), followed by SMP2SMP. On this basis PESTPRP1 prepares an instruction file to read the SMP2SMP output file, and builds a PEST control file containing measured values as listed in the "measurement" site sample file on which the model-generated site sample file produced by SMP2SMP is based; parameters listed in a set of template files are also recorded in the PEST control file generated by PESTPRP1.

Note that, with the advent of TSPROC, the role of PESTPRP1 in conjunction with SMP2SMP has now been largely usurped by this considerably more powerful program.

Using PESTPRP1

PESTPRP1's use as a PEST pre-processor is predicated on the assumption that the model run by PEST produces as one of its output files a site sample file in which model outputs are time-interpolated to measurement dates and times. Recall that SMP2SMP writes a site sample file containing model-generated quantities interpolated to the times at which field measurements were made, the latter being supplied in a "measurement" site sample file. It is the role of PESTPRP1 to write an instruction set by which the SMP2SMP-generated site sample file can be read, and to generated a PEST control file whose corresponding observation values are extracted from the "measurement" site sample file.

PESTPRP1 begins execution with the prompts:-

```
Enter name of observation site sample file:
Enter name of    model    site sample file:
```

The first is the site sample file containing measured data. The second is the site sample file generated by SMP2SMP as part of the model. Note that it is essential that the above site sample files be "paired" in the sense that the latter is generated by SMP2SMP on the basis of the former as part of the composite model run.

As already stated, PESTPRP1 writes an instruction set by which the model-generated site sample file can be read, as well as a PEST control file. The production of both of these files entails the generation of observation names. PESTPRP1 generates observation names in one of three ways, depending on the user's choice. PESTPRP1 prompts:-

```
Use numbers or site identifiers for observation names?  [n/s]:
```

If "n" is selected, observations are named from 1 to 99999999 in order of their appearance in the SMP2SMP-generated site sample file (which will also be the order of their appearance in the measurement site sample file upon which the SMP2SMP-generated site sample file is based). Alternatively, select "b" for greater ease in relating observation names to actual measurements. In that case PESTPRP1 prompts:-

```
Use first n or last n characters of site identifier?  [f/l]:
```

where *n* is a number from 3 to 6. If "f" is typed in response to the above prompt, PESTPRP1 generates observation names by taking the first *n* characters of the site identifier and affixing the suffix "_*m*" to the identifier's name, where *m* signifies the *m*'th sample pertaining to that site as read from the SMP2SMP-generated site sample file. PESTPREP determines *n* in the above prompt through counting the maximum number of observations pertaining to any site and thus determining how many of the twelve characters available in an observation name can be assigned to sample numbering in this fashion. If this method of assigning observation names does not result in a unique set of names due to the fact that different site identifiers have the same first *n* letters in common, PESTPRP1 informs the user of this. He/she is then prompted for an alternative method of observation name generation.

If the response to the above prompt is "l", PESTPRP1 uses the last *n* characters of each site identifier in conjunction with the measurement sequence numbering scheme to determine observation names. Once again, if this methodology does not result in a set of unique observation names PESTPRP1 will not proceed, requesting instead that the user employ an alternative scheme for observation name generation.

PESTPRP1's next prompt is:-

```
Enter name for instruction file:
```

Once it is supplied with this name (preferably with an extension of *.ins*) PESTPRP1 generates the instruction set by which the SMP2SMP-generated site sample file can be read. Then it gathers the names of the various parameters involved in the current parameter estimation problem by reading all template files involved in the current PEST run. It prompts:-

```
How many template files are there?
Enter name for template file # 1:
Enter name for template file # 2:
```

Once it has read these template files it prompts for the name of the PEST control file which it must write:-

```
Enter name for output PEST control file:
```

which it then proceeds to write.

When generating the "parameter data" section of the PEST control file, PESTPRP1 assigns all parameters an initial value of 1.0 and provides lower and upper bounds of -10^{10} and 10^{10} respectively. A default transformation type or "none" is assumed. This will almost certainty require manual editing in order to provide values which are more appropriate for the current parameter estimation problem. Note in particular that logarithmic transformation of certain parameter types is often crucial to the success of the parameter estimation process.

In building the "parameter groups" section of the PEST control file, PEST assigns each parameter to a group of its own and supplies certain default values which control the calculation of derivatives. This user should be aware of the need to assign more appropriate values in many circumstances.

PESTPRP1 assumes that only one instruction file is required by the inversion process, this being the one that it has written itself to read the SMP2SMP-generated site sample file, of which it also knows the name. PESTPRP1 also knows the names of all template files involved in the inversion process (but not the names of the corresponding model input files). It records all of this information to the PEST control file. Once this has been written, the status of the PEST input dataset can be immediately checked using program PESTCHEK. However, at the very minimum, the user will need to alter the model command line (which PESTPRP1 assumes to be simply *model*) and the names of the model input files corresponding to the various template files (which PESTPRP1 has temporarily named *model1.in*, *model2.in*, etc) before actually running PEST.

PESTPRP1 assumes that PEST will run in parameter estimation mode. It adds no "predictive analysis" or "regularisation" section to the PEST control file which it generates. Likewise, it records no prior information. If required, both of these can be easily added by the user.

Note that when using PESTPRP1, or any other member of the PEST Surface Water Utilities suite, the user can "backtrack" in execution by responding to any prompt by simply pressing "e" followed by the <Enter> key; "e" stands for "escape".

Uses of PESTPRP1

PESTPREP automates most of the laborious work required in preparing for a PEST run. Because it can rapidly process the large amounts of data that often accompany model calibration, it can accomplish in seconds that which would take hours to accomplish in any other way.

See Also

See also SMP2SMP and TSPROC.

PLT2SMP

Function of PLT2SMP

PLT2SMP generates a site sample file on the basis of a "plotting file" written by the PLTGEN module of HSPF. This makes HSPF results accessible for processing by various members of the Surface Water Utility programs documented herein. It also allows HSPF results to be directly compared with field data by running HSPF followed by PLT2SMP followed by SMP2SMP. Calibration of HSPF can then be undertaken using PEST in conjunction with a composite model comprised of these three programs.

Note however that, with the advent of TSPROC, the role of PLT2SMP has now been largely superseded by this considerably more powerful program.

Using PLT2SMP

Because PLT2SMP uses dates and times, it is essential that a settings file named *settings.fig* reside in the subdirectory from which PLT2SMP is run. Recall that the contents of this file inform any utility which reads it of the format to use when reading and writing dates, in particular whether the day precedes the month or vice versa. If a settings file is not present in the subdirectory from which PLT2SMP is run, PLT2SMP terminates execution with an appropriate error message.

PLT2SMP commences execution with the prompt:-

```
Enter name of PLTGEN-generated HSPF output file:-
```

Supply an appropriate filename in response to this prompt. PLT2SMP then reads the header to this file, extracting the following information:-

- the number of time series (ie. curves) represented in this file, and
- the labels associated with these time series.

It then presents each of these to the user and asks whether these should be reproduced in the site sample file to be written by PLT2SMP. If so, a "site identifier" is required for each such time series. PLT2SMP asks the user for such an identifier if a series is to be transferred, supplying as a default identifier the first 10 characters of the HSPF time series label (with underscores substituted for blanks if appropriate). In each case the user may either accept the default by pressing the <Enter> key, or supply a label him/herself.

After PLT2SMP has prompted in this fashion for all time series represented in the PLTGEN file, it prompts for the name of the site sample file which it must write. After being supplied with this name it writes the file, informing the user when the task is complete.

When writing dates and times to the site sample file, PLT2SMP makes a slight alteration to the HSPF representation of some dates and times. If a time series value corresponds to

midnight, HSPF represents the time as 24:00:00 and the date as that of the preceding day. However the convention adopted by the Surface Water Utilities is that the time be represented as 00:00:00 on the following day. PLT2SMP makes the appropriate adjustment where necessary.

Uses of PLT2SMP

PLT2SMP is used to translate HSPF-generated data into a format where it is accessible by programs of the Surface Water Utilities for further processing.

A particularly useful application of PLT2SMP is the building of a site sample file as part of a composite model comprised of HSPF, PLT2SMP and SMP2SMP. Outputs of such a model are model-generated quantities time-interpolated to the times at which corresponding field measurements were made, thus allowing the two sets of data to be directly compared. Such a composite model is readily used in conjunction with PEST for HSPF calibration. You should note, however, that if versions of HSPF prior to version 12 are to be used with PEST in this fashion, it would be very wise to adjust the HSPF source code slightly in order to ensure that the PLTGEN module writes numbers to its output files with the full number of available significant figures. If you cannot do this yourself, contact Watermark Computing for a copy of HSPF in which this has already been done.

See Also

SMP2SMP.

SMP2HYD

Function of SMP2HYD

SMP2HYD reads a site sample file. For each member of a list of user-specified sites, SMP2HYD extracts all of the information pertinent to those sites within a user-specified time window. It then rewrites this information to a series of output files (one for each site) in a form fit for immediate use by scientific graphing software.

Using SMP2HYD

Program SMP2HYD will not run unless a settings file (*settings.fig*) is present within the directory from which it is invoked. As discussed in the introduction to this manual, a settings file determines the manner in which dates are represented by the Surface Water Utilities.

SMP2HYD begins execution with the prompt:

```
Enter name of site sample file:
```

to which you should respond by typing an appropriate filename.

Next SMP2HYD prompts for the names of the sites for which time-dependent information is required, and for the files to which it should write this information.

```
Enter sites for which hydrographs are required (Press <Enter> if no more):-
  Enter site for hydrograph number 1:
  Enter output file for hydrograph number 1:
  Enter site for hydrograph number 2:
  Enter output file for hydrograph number 2:
                    .
                    .
```

Enter, in response to the first of each pair of these prompts, a site identifier. In each case the site should be featured in the site sample file whose name was provided earlier. Press <Enter> when you wish to supply the identifiers for no further sites. In response to the second of each pair of prompts supply a filename to which SMP2HYD should write the information which it extracts from the site sample file for the site whose identifier was provided in response to the first prompt of the pair.

When there are no further sites SMP2HYD asks:

```
Use all samples for nominated sites, or specify sample window?  [a/w]:
```

If you enter “a” in response to the above prompt, SMP2HYD will extract from the site sample file the entirety of the information found in that file for each of the sites supplied in response to the preceding prompts. However if you respond with “w”, SMP2HYD will extract information for each site only within a time window whose details must be supplied next in response to the prompts:

```
Enter sample window start date  [dd/mm/yyyy]:
Enter sample window start time    [hh:mm:ss]:
Enter sample window finish date [dd/mm/yyyy]:
```

```
   Enter sample window finish time   [hh:mm:ss]:
```

Note that SMP2HYD will employ a *mm/dd/yyyy* format for date representation if dictated by the settings in file *settings.fig*.

Because SMP2HYD writes an output file in which site measurements are recorded against elapsed time, it needs to know the reference time from which elapsed time is measured. So it asks:

```
 When is zero time?
   Enter reference date [dd/mm/yyyy]:
   Enter reference time   [hh:mm:ss]:
```

Next SMP2HYD prompts for the units in which it should express elapsed time on its output file:

```
 Enter output time units (yr/day/hr/min/sec) [y/d/h/m/s]:
```

Having now acquired all of the data that it requires, SMP2HYD reads the site sample file, extracting information for those sites for which this information was requested, and writing it to the output file nominated for that site. The figure below shows part of such a SMP2HYD output file.

TIME_IN_DAYS	DATE	TIME	SITE_13500006A
-490.50	02/12/1965	12:00:00	18.5200
-435.50	26/01/1966	12:00:00	18.5000
-434.50	27/01/1966	12:00:00	18.5000
-434.50	27/01/1966	12:01:01	18.0000
-402.50	28/02/1966	12:00:00	18.4500
-371.50	31/03/1966	12:00:00	18.4300x
-322.50	19/05/1966	12:00:00	18.3000x
-300.50	10/06/1966	12:00:00	18.3300x
-275.50	05/07/1966	12:00:00	18.1500
-182.50	06/10/1966	12:00:00	18.1200
-119.50	08/12/1966	12:00:00	18.0000
-048.50	17/02/1967	12:00:00	17.9200x
-31.50	06/03/1967	12:00:00	17.9200x
1.50	15/05/1967	12:00:00	17.9200x
24.50	21/06/1967	12:00:00	18.5600
102.50	13/07/1967	12:00:00	18.5300
175.50	09/08/1967	12:00:00	18.6300
217.50	06/10/1967	12:00:00	18.6300
257.50	05/12/1967	12:00:00	18.5600
323.50	08/01/1968	12:00:00	18.5600
360.50	11/03/1968	12:00:00	19.2900
393.50	17/05/1968	12:00:00	19.4700

Extract from a SMP2HYD output file.

The first column of a SMP2HYD output file lists time elapsed since the reference time; for samples prior to the reference time elapsed times are negative. (Note that the header to this column records the units used for elapsed time as previously supplied by the user.) The second and third columns list sample dates and times respectively; these were transferred directly from the site sample file (reset file *settings.fig* to record dates in the *mm/dd/yyyy* format). The fourth column lists the sample value. Note that if a value was marked by an "x" in the site sample file, the "x" is transferred to the SMP2HYD output file. However instead of being in a column of its own, it is placed directly against the number which it denotes as suspect. This makes the sample value an invalid number. Some plotting packages, when they fail to read the number, will object with an error message; others will simply ignore the

number, probably a desirable feature when the data is plotted. A user can search for the presence of suspect data in a SMP2HYD output file by simply importing the file into a text editor and searching for "x".

The header to the values column of a SMP2HYD output file records the name of the site to which the values pertain.

Uses of SMP2HYD

For cases where a site sample file has been created by exporting data from a site database, SMP2HYD provides the means whereby hydrographs can be generated from that data with the maximum flexibility. A site sample file can also be created by a model; see, for example programs IQQM2SMP and PLT2SMP; the LIST_OUTPUT block of TSPROC can also write files which, with only minor editing, conform to site sample file specifications. In these cases SMP2HYD provides a mechanism for the plotting of model-generated data as well. Once again, by using a commercial scientific graphing package to undertake this plotting, graphs can be constructed with maximum flexibility.

See Also

See also programs IQQM2SMP, PLT2SMP, SMPCHEK and TSPROC.

SMP2SMP

Function of SMP2SMP

Although it can be used in many situations, SMP2SMP was designed for use in a model calibration context. It is assumed that the outcome of a model run is a site sample file in which model-generated outcomes at a number of points, or of a number of different kinds, are listed together with the dates and times to which they pertain. Normally such model-generated data will be available at a large number of dates and times distributed regularly, or semi-regularly, through the model simulation time.

It is also assumed that another site sample file is available, this file containing field observations of certain quantities within the model domain. This file may contain measurements at locations, and of types, not cited in the model-generated site sample file. It will almost certainly contain samples at times which differ from model output times; some of these times may pre-date the commencement of the model simulation, while others may postdate the model simulation time span.

SMP2SMP reads both the model-generated and observed site sample files. It produces a third site sample file by time-interpolating model results to the times and dates of field measurements for measurement types that are represented in both files; measurement types are recognised as being equivalent if they possess the same site identifier. Thus the outcome of SMP2SMP's execution is a site sample file containing model-generated data interpolated to field measurement times, thereby allowing a ready comparison to be made between field and model-generated data. However the site sample file produced by SMP2SMP is likely to be shorter than the observation site sample file, as measurement types not represented in the model-generated site sample file are omitted. Measurement dates and times either before or after the model simulation time span are also omitted as interpolation cannot take place to these dates and times.

If SMP2SMP is run as part of a composite model, the "model-output file" generated by it is amenable to processing by PESTPRP1 (also documented within this manual). Thus preparation for a parameter estimation run using PEST becomes a trivial task.

Note that most of the functionality available through SMP2SMP is also available through the newer, more powerful, TSPROC.

Using SMP2SMP

A settings file (named *settings.fig*) must be present within the subdirectory from which SMP2SMP is run. Depending on the contents of this file, dates are assumed to be represented either in the format *dd/mm/yyyy* or *mm/dd/yyyy* in all site sample files processed and produced by SMP2SMP.

SMP2SMP begins execution with the prompt:-

```
Enter name of observation site sample file:
```

The user should respond with the appropriate site sample filename.

SMP2SMP then prompts for the name of a model-generated site sample file:-

```
Enter name of model-generated site sample file:
```

in response to which an appropriate filename should be supplied.

The following points should be noted regarding both the observation and model-generated site sample files:-

- Both of these files should be checked for errors and inconsistencies using program SMPCHEK prior to being supplied to SMP2SMP.

- It is not necessary that one site sample file contain observation data and the other contain model-generated data. Though this will often be the case, these descriptions are used within the present context to differentiate between the two different files.

- The two site sample files should have at least some site identifiers in common, for this is the variable that SMP2SMP uses to link data types in one file to those in the other. Note that site identifiers are case insensitive.

SMP2SMP next prompts:-

```
Enter extrapolation threshold in days (fractional if necessary):
```

SMP2SMP carries out linear temporal interpolation between model output times as represented in the model-generated site sample file, to measurement times as represented in the observation site sample file; linear interpolation to a measurement time takes place on the basis of two model output times, one on either side of the measurement time. However if the measurement time precedes the first model output time for a particular measurement type, or postdates the last model output time, then SMP2SMP will assign a data value to that time equal to the first or last model sample if the measurement time is within *x* days of the beginning or end of the model simulation time, *x* being the user's response to the above prompt. If desired, *x* can be less than a day, or even zero.

Finally SMP2SMP prompts for the name of the site sample file which it must generate. After having been supplied with this name, it searches for site identifiers represented in the observation site sample file which are also represented in the model-generated site sample file. If any of the samples pertaining to these identifiers fall within the model simulation time window, SMP2SMP interpolates model results to the dates and times corresponding to the samples. It then writes a new site sample file containing model-generated equivalents to field observations.

As was mentioned above, the observation site sample file can contain measurements outside of the model simulation time span, and can reference sites (or measurement types) that are not cited in the model-generated site sample file. In neither case is a corresponding model-generated sample represented in the SMP2SMP-produced site sample file. Also, if a sample in the observation site sample file is accompanied by an "x" in the final column indicating suspect data (see Appendix B of this manual), then SMP2SMP does not interpolate model

results to this sample. In the unlikely event that a model-generated sample is "x-affected", that sample is not used in the temporal interpolation process; the preceding sample or the next sample is used instead.

At the end of its execution SMP2SMP lists to the screen the names of sites which are represented in the observation site sample file, but which are not represented in the model-generated site sample file (if any such sites exist). It also lists the names of sites for which all observation samples fall outside the model simulation time window.

Uses of SMP2SMP

SMP2SMP is particularly useful in model calibration. By including SMP2SMP as part of a composite model encapsulated in a batch file, the model is able to generate model outputs at the exact times at which there are field measurements. Thus a direct comparison between the two can be made. If model calibration is undertaken using PEST, program PESTPRP1 (also documented in this manual) can be run in order to automate the building of PEST input files; through this mechanism the time required for PEST setup can be reduced to minutes even when calibrating complex models.

SMP2SMP is used where a particular model executable program or post-processor produces a site sample file listing model results at model output times. Thus SMP2SMP can be run following, for example, programs PLT2SMP and IQQM2SMP as part of a composite model.

See Also

PESTPRP1 and TSPROC.

SMP2VOL

Function of SMP2VOL

SMP2VOL reads a site sample file; it is assumed that values recorded in this site sample file represent flows. For any of the sites listed in this file, SMP2VOL is able to calculate the total flow volume between two arbitrary dates and times; these dates and times need not correspond to sample dates and times.

Note that most of the functionality available through SMP2VOL is also available through the newer, more powerful, TSPROC.

Using SMP2VOL

Usage of SMP2VOL requires that a setting file named *settings.fig* be present in the directory from which it is run. As is explained in the introduction to this manual, the contents of this file inform SMP2VOL of the date format being used.

Upon commencement of execution SMP2VOL prompts for the name of a site sample file containing flows recorded at one or more sites. As is the case for all programs which use them, the site sample file should be checked for data integrity using program SMPCHEK before use by SMP2VOL.

After checking for the presence of this file and then opening it, SMP2VOL prompts:-

```
Enter time units used for flow in this file [s/m/h/d]:
```

Allowed units are seconds, minutes, hours and days respectively. Volumes calculated by SMP2VOL are expressed in the same volume units as those used for flow in the site sample file which it reads; SMP2VOL does not need to know these units.

SMP2VOL next prompts for the name of a "dates file". This is a file that must be prepared by the user prior to running SMP2VOL. Its format is shown below:-

```
guage_1   23/04/1989  12:00:00  01/09/1993  12:00:00
guage_1   30/06/1990  12:00:00  30/07/1990  15:00:00
guage_4   03/04/1970  00:00:00  03/04/1990  12:00:00
guage_1   12/12/1996  12:00:00  12/12/1996  19:00:00
etc
```

Part of a dates file.

Each line of a dates file must contain five entries. The first entry is the name of a site listed in the site sample file. The next two entries are the date and time corresponding to the beginning of the interval over which volume is to be calculated for that site; the last two entries are the date and time corresponding to the end of the volumetric calculation interval.

Note the following points concerning a dates file:-

- the second date and time must postdate the first date and time on each line of the file;
- the site comprising the first entry of each line must correspond to a site listed in the site sample file;
- site entries can be in any order in the dates file; they need not correspond to the ordering of entries in the site sample file; nor do references to any one site need to be together;
- time intervals represented on different lines of the dates file can overlap, predate or postdate intervals represented on other lines of the file.

After it has read the dates file (and checked it for any errors, the presence of which will be reported immediately to the screen), SMP2VOL prompts for the name of an output file to which it will write the outcomes of its volumetric calculations. Once this has been supplied, SMP2VOL writes this file and terminates execution.

The SMP2VOL output file has identical format to the user-supplied dates file. However a sixth column is added in which volumetric calculations over the nominated intervals are recorded. These are written with 7 significant digits of precision in case of PEST usage based on this file (see below). However if, for some reason, the requested volume cannot be calculated, SMP2VOL writes a text string in place of the number, this string explaining the reason for absence of the number. Reasons why volumetric calculation cannot take place include the following:-

- a site listed in the dates file is not cited in the site sample file,
- the beginning of the volumetric calculation interval predates the earliest sample for a particular site,
- the end of the volumetric calculation interval postdates the latest sample for a particular site,
- all flow readings for a particular site are of questionable integrity (denoted by an "x" in the final column of the site sample file – see Appendix B for further details).

Uses of SMP2VOL

Apart from its obvious application of allowing accumulation of flow rates over arbitrary time intervals in order to calculate volumes, SMP2VOL can form a useful component of a composite model run by PEST. Stability of a model calibration exercise undertaken by PEST in which model-generated flows are matched to observed flows can often be enhanced if volumetric data is used in conjunction with flow data in the inversion process. Using this methodology, one or a number of volumetric observations can be added to the flow time series (with appropriate weights). "Measured" flow volumes can be calculated on the basis of a field-observation site sample file, whereas model-generated flow volumes can be computed by running SMP2VOL as part of the composite model to calculate volumes on the basis of a model-generated site sample file. The latter may be produced by programs such as PLT2SMP and IQQM2SMP which would also comprise part of the composite model. An even better

idea is to compute "modelled volumes" on the basis of a SMP2SMP output file (SMP2SMP being run as part of the composite model). If this is done, volumetric calculation will be carried out on the basis of flows pertaining to exactly the same times in both observation and modelled datasets. Any interpolation errors will then effect both of the calculated volumes equally.

See Also

IQQM2SMP, PLT2SMP and TSPROC.

SMPCAL

Function of SMPCAL

Program SMPCAL is used to calibrate one dataset against another. In most cases data requiring adjustment will be that gathered by an electronic logging device (for example a pressure sensor or flow meter) while data used for calibration will consist of a number of manual readings taken over the time period during which the logger was operating.

Using SMPCAL

Configuration Files

As soon as SMPCAL commences execution it searches the current directory for a file named *settings.fig* in order to ascertain the protocol which it must use to represent dates. See the introduction to this manual for further details.

Site Sample Files

SMPCAL must be supplied with the names of two site sample files. In most cases these site sample files can be easily downloaded from a user's database.

Each of the site sample files supplied to SMPCAL can be comprised of data from one or many sites. It is presumed that one of these site sample files contains "raw" data (usually logger data) for which sampled values need to be adjusted against data contained in another site sample file containing "true" or "standard" readings (for example manually-gathered data). In the discussion that follows the site sample file containing data that requires adjustment is referred to as the "logger" site sample file; the file containing data against which this adjustment takes place is referred to as the "standard" site sample file.

The following points regarding the contents of the two site sample files supplied to SMPCAL should be noted.

- Every site cited in the logger site sample file should also be cited in the standard site sample file; the converse is not the case.

- For each site cited in the logger site sample file, there should be at least two samples in the standard site sample file within the time frame spanned by the first and last readings for that site in the logger site sample file (or just slightly outside of that time span - see below).

- Both site sample files supplied to SMPCAL should obey all rules pertaining to the construction of a site sample file; it is a good idea to check them both using program SMPCHEK before processing them with SMPCAL.

What SMPCAL Does

SMPCAL evaluates constants *m* and *c* by which logger data may be adjusted to agree with standard measurements of the same quantities using the equation:

$$d_s = m\, d_l + c$$

where d_l is logger data and d_s represents standard data. SMPCAL calculates a different value of *m* and *c* for every interval between standard measurements within the standard site sample file. When adjusting logger data, this *m* and *c* is then applied to all logger measurements taken between the two standard samples used in their derivation. For logger samples preceding the first standard sample, the *m* and *c* determined for the first standard interval are employed in data adjustment. For logger samples post-dating the last standard sample, the *m* and *c* determined for the last standard interval are employed in data adjustment.

Where a logger sample time does not coincide with a standard sample time, logger data (normally more closely spaced than standard data) is linearly interpolated to the time of the standard measurement to determine a notional logged value at that time for the purposes of determining *m* and *c*. Where a standard sample precedes or post-dates the first logger sample, logger readings can be linearly extrapolated (using the first or last two logger samples) to the standard sample time for this same purpose.

It is worth noting that the interpolation scheme used by SMPCAL to obtain notional logged values at standard measurement times is actually a linear extrapolation process using the two sample values either before or after the standard measurement time, depending on whether calibration coefficients are being sought for the preceding or following logged interval. The reason for this is that, for some logging systems, the downloading of logger data (which often accompanies manual measurement) results in an unfortunate "glitch", or offset, in logged values at the time at which these values are downloaded. By undertaking individual extrapolation from either side of the standard measurement point, the effect of this extraneous offset can be "calibrated out".

Running SMPCAL

SMPCAL is run using the command:

```
smpcal
```

It requires only five items of information, the first two of which are the names of the logger and standard site sample files. The prompts are as follows:-

```
Enter name of site sample file requiring calibration:
Enter name of standard site sample file:
```

Note that, in common with other programs from the Surface Water Utilities, the user can "backtrack" at any time to the previous prompt by replying with an "e" (for "escape") to any particular prompt. Note also that, in accordance with the specifications of a site sample file as set out in Appendix B of this manual, an "x" in the 5th column of a site sample file signifies dubious data in the 4th column. Such lines are ignored by SMPCAL, being used neither for

data calibration (if occurring in the standard data file) nor for data adjustment (if occurring in the logger data file).

Next SMPCAL prompts:

```
Enter maximum extrapolation time in hours:
```

All standard samples lying within a time period beginning *h* hours before the first logged sample for a particular site and *h* hours after the last logged sample for that site (where *h* is supplied in response to the above prompt) will be used in the data calibration process (ie. the process of determining *m* and *c).* Next SMPCAL prompts for the names of its output files. First the site sample file which it generates by multiplying each logged data value by an appropriate *m* and *c*:

```
Enter name for calibrated site sample file:
```

Then its run record file:

```
Enter name for report file:
```

The figure below shows part of a SMPCAL report file. A record similar to that shown in the figure is presented for each site cited in the logger site sample file. Note that the string "`not used`" depicts the case where adjacent standard samples both lie between neighbouring logger samples, or where two adjacent standard samples both pre- or postdate all logger samples. In neither of these cases is an *m* or *c* value required for the adjustment of any logger data.

```
Data adjustment for site SA123:-


  Raw data ---->
       First sample of raw data at:      31/12/1996  14:00:00
       Last  sample of raw data at:      12/02/1997  16:00:00
       Total number of samples: 512

   Standard data within calibration time frame ---->
       First sample of standard data at: 31/12/1996  13:25:00
       Last  sample of standard data at: 12/02/1997  14:30:00
       Total number of samples: 6

   Calibration equation  -   Y = M*X + C  ---->

                      Interval                          M           C
   ------------------------------------------------------------------------
    31/12/1996 13:25:00 to 06/01/1997 13:20:00      2.1323E-03  -16.21
    06/01/1997 13:20:00 to 13/01/1997 13:25:00      2.1738E-03  -16.27
    13/01/1997 13:25:00 to 20/01/1997 13:25:00        not used
    20/01/1997 13:25:00 to 29/01/1997 13:07:00      2.1787E-03  -16.29
    29/01/1997 13:07:00 to 12/02/1997 14:30:00      2.2019E-03  -16.36
```

Part of a SMPCAL report file.

If, during its execution, SMPCAL encounters a problem with either of its input site sample files it writes an appropriate error message to the screen and terminates execution. Normally only a single message is written, this being clearly visible upon termination of SMPCAL execution. However for certain types of error, SMPCAL continues execution until all of the

logger sample file is processed, continuing to report further errors to the screen as necessary. To inspect all errors generated in this way, place the answers to the five SMPCAL prompts in a file (for example *smpcal.in*), one under the other, and run SMPCAL using the command:

```
smpcal < smpcal.in > temp.dat
```

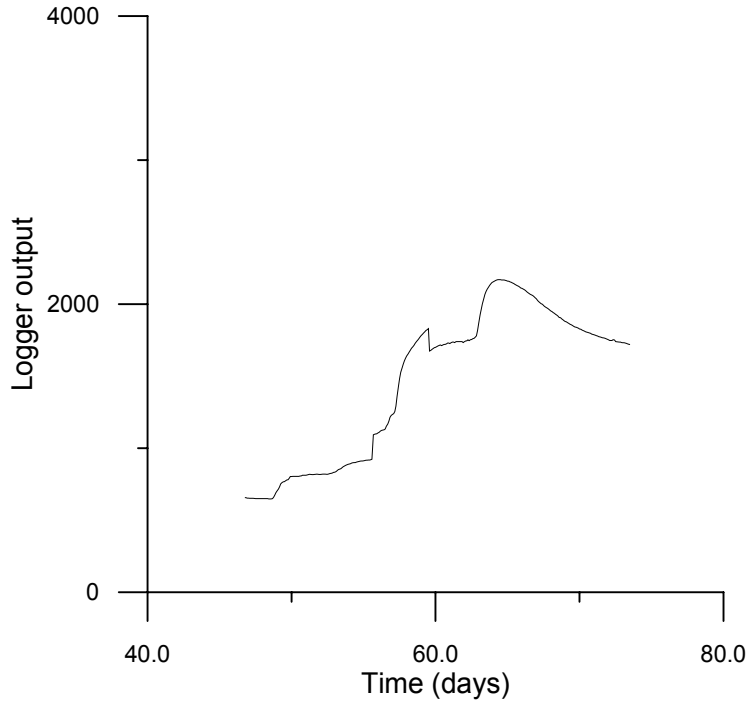
In this case SMPCAL takes its input from file *smpcal.in* and writes its normal screen output to a file named *temp.dat* which can be inspected at leisure.

It is a general principal of data calibration that standard samples should span as wide a data range as possible. As is documented below, if this procedure is not followed data adjustment using SMPCAL can lead to unpredictable results. If neighbouring standard points have identical value, the calibration process breaks down altogether and SMPCAL reports an appropriate error message to the screen. It will also report an error to the screen if logger readings, as interpolated to neighbouring standard readings, are identical, even if the manual readings are not, for then *m* is assigned the impossible value of zero.

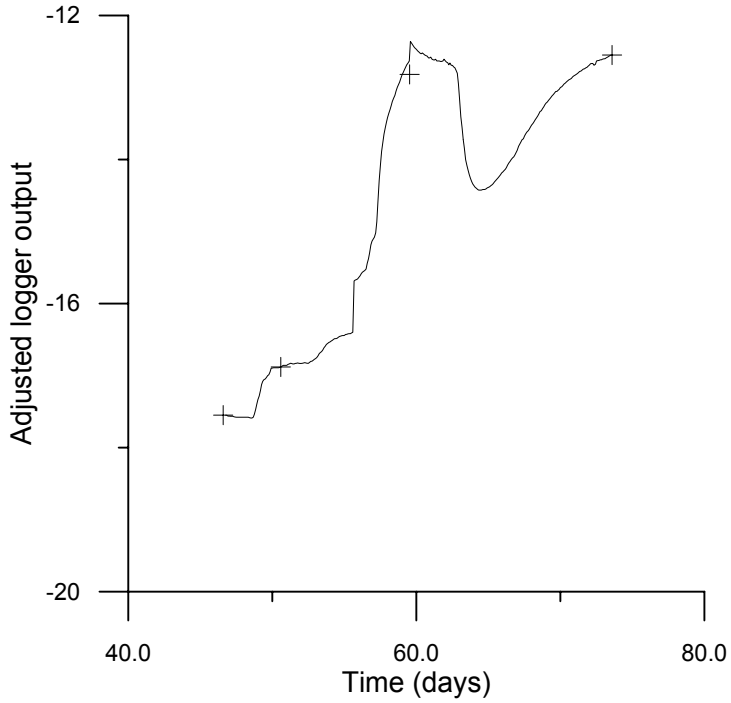
Uses of SMPCAL

The most obvious use of SMPCAL is to calibrate logger data against manual readings taken over the time that the logger was operating. The fact that *m* and *c* (as determined by SMPCAL) can vary with time facilitates automatic logger data adjustment to real measurement datum, even where logger calibration drifts with time. However this can also lead to serious errors in data adjustment if the user is not extremely careful, as the following example demonstrates.

The figure below shows a graph of raw logger data plotted against time. Following that is another graph, this one showing corrected logger data. Corrections were made on the basis of the four manual readings depicted in that figure.



Raw logger data plotted against time.



Adjusted logger data plotted against time; adjustment carried out on basis of the four points shown.

It is obvious from the second of these two graphs that there is something seriously wrong with the data adjustment process, for the peak occurring in the latter half of the logged

interval has been converted to a trough. The reason for this is as follows. As can be seen from the first graph, a slight "glitch" occurs in logger readings just before 60 days, this being shortly after the second last manual reading was taken. As a result of this glitch, all subsequent logger output data is displaced downwards. The two manual readings that were used to derive *m* and *c* for the final interval span this glitch. To make matters worse, the levels recorded for these two manual readings are very similar, the vertical separation of these readings being of lower magnitude than the size of the glitch. Unfortunately there is a considerable logged level change within the last data interval; hence all logged readings comprising this event had to be adjusted using an *m* and *c* calculated on the basis of two manual readings whose vertical separation was minimal. This, compounded by the glitch of larger magnitude than the vertical separation of the standard sample points, resulted in an *m* of opposite sign to that calculated for other manual data point pairs, with the result that positive variations in logger output resulted in negative movements of the adjusted dataset.

Other possible pitfalls are many. For example a peak in logger response can be amplified or attenuated if calibration points of approximately equal data value are situated on either side of it. This is a result of erroneous determination of *m* because of measurement inaccuracies or the limited number of significant figures used by the logger.

Hence the user must exercise considerable caution when using SMPCAL. In particular, the following points must be observed.

- always plot both the raw logger data and adjusted data,
- inspect the SMPCAL report file; significant variations of *m* and *c* between measurement data pairs is a good indicator or an unreliable calibration,
- it is best to use only a few standard data points spanning as high a vertical interval (ie. range of data values) as possible. **Neighbouring standard points should not have similar data values.**
- if possible, it is good practice to take two manual readings when visiting a site, one before downloading logger data and one after, in case the process of downloading the logger is responsible for any glitches.

Plotting of data within a site sample file can be easily accomplished using program SMP2HYD together with a commercial plotting package. If GRAPHER is used in the latter capacity, then updated site sample files generated on subsequent SMPCAL runs can be easily viewed by writing subsequent SMP2HYD plotting data to files of the same name; SMP2HYD's operations can be automated through writing its input data to a file and supplying this data to SMP2HYD using input file re-direction.

When plotting adjusted data together with standard data as in the second of the above figures, standard data points will plot on the adjusted data curve. However the coincidence will not always be exact for, as has been mentioned above, where the time of a standard sample does not coincide with the time of a logged sample, the logged time series is interpolated to the time of standard measurement. If there are rapid logger variations in the immediate vicinity of a manual reading (such as the glitch depicted above), the appearance that the standard reading does not plot exactly on the adjusted data results from the fact that the "join" between

two neighbouring data intervals (in which different *m* and *c* values were used for adjustment) lies between two logged data points. The plotting package used to plot the adjusted data uses straight lines to join neighbouring logged data points; it does not deviate to account for the notional data value at the joining point. The latter, if it were determined by extrapolation from each neighbouring data time interval, would indeed plot at the same location as the standard point.

See Also

See also SMP2HYD.

SMPCHEK

Function of SMPCHEK

SMPCHEK checks the integrity of a site sample file, reading it in its entirety and reporting any errors that it finds to the screen.

Running SMPCHEK

Upon commencement of execution, SMPCHEK prompts:

```
Enter name of site sample file:
```

Respond to this prompt by typing the name of an appropriate site sample file.

SMPCHEK then reads the site sample file, checking:

- that every line of the file has sufficient entries,
- that all numbers, dates and times are readable,
- that all dates and times are legal,
- that site identifiers are 10 characters or less in length, and
- that all entries for each site are consecutive and are in order of increasing date and time.

SMPCHEK writes any errors it detects to the screen (redirect screen output to a file for a more permanent record). Each error message includes a line number, allowing a user to locate the error and rectify it.

Uses of SMPCHEK

Though many of the PEST Surface Water Utilities documented in this manual read a site sample file, none of them check the file for errors to the same extent that SMPCHEK does. Furthermore if an error is detected by one of these programs, execution is often aborted after the error has been detected and reported; thus other errors are left undetected, only to be reported in later processing after errors closer to the top of the site sample file have been rectified. SMPCHEK was written to overcome the inconvenience of detecting and reporting errors in a piecemeal manner by programs that were written to perform other tasks. SMPCHEK can detect and report all of the errors in a site sample file at once. (However it is configured to report only the first 40; if there are more than 40 errors there is probably something seriously wrong with the formatting or layout of the site sample file.)

Once a site sample file has been created (either by directly downloading a file from a database or by performing some elementary processing of a database-downloaded file), the

file should be checked with SMPCHEK to establish its integrity. Errors can then be corrected so that use of the file in subsequent processing will hold no surprises.

See Also

See also SMP2HYD, SMP2SMP, SMP2VOL.

Appendix B

File Formats

File Formats

Site Sample File

The "site sample file" is fundamental to the operation of many of the Surface Water Utilities; it holds time series data gathered at one or a number of sites. The data stored in this file can be of any type.

A site sample file records data gathered at discrete sample times at a number of specific locations, eg. water level or chemical concentration data gathered through sampling programs. Each line of a site sample file has four (or possibly five) entries, each of which must be separated from its neighbouring entry by one or more white space (including tab) characters. Typically a site sample file will hold data extracted from a database. Part of a site sample file is shown below.

```
  13500002A    25/09/1991   12:00:00  12.00
  13500002A    02/01/1992   12:00:00  11.83
  13500002A    24/03/1992   12:00:00  12.81
  13500002A    29/06/1992   12:00:00  13.54
  13500002A    22/09/1992   12:00:00  13.24
  13500002A    17/12/1992   12:00:00  12.84
  13500002A    22/03/1993   12:00:00  12.38 x
  13500002A    21/06/1993   12:00:00  11.83 x
  13500002A    27/09/1993   12:00:00  11.61 x
  13500002A    16/12/1993   12:00:00  12.35
  13500002A    01/03/1994   12:00:00  11.79
  13500002A    22/03/1994   12:00:00  11.89
  1351235A     19/02/1959   12:00:00  29.84
  1351235A     05/03/1959   12:00:00  30.33
  1351235A     20/03/1959   12:00:00  30.76
  1351235A     06/04/1959   12:00:00  31.19
  1351235A     17/04/1959   12:00:00  31.45
  1351235A     01/05/1959   12:00:00  31.65
  site_a       15/05/1959   12:00:00  31.65
  site_a       29/05/1959   12:00:00  31.65
  site_a       12/06/1959   12:00:00  31.65
  site_a       26/06/1959   12:00:00  31.46
  site_a       10/07/1959   12:00:00  31.34
```

Extract from a site sample file.

The first item on each line of a site sample file is a site identifier. This identifier must be of 10 characters or less in length. When used with programs of the Surface Water Utilities the site identifier is case-insensitive. The second item is the date; depending on the contents of the settings file *settings.fig* (see the introduction to this manual), this must be expressed either in the format *dd/mm/yyyy* or *mm/dd/yyyy*. Then follow the time (in the format *hh:mm:ss*) and the measurement pertaining to the cited date and time. An optional fifth item may be present on any line; if present, this item must consist solely of the single character "x" to indicate that the previous data element lacks integrity.

The following rules must be observed when generating a site sample file:

- For any one site dates and times must be listed in increasing order.

- All entries for the same site must be in juxtaposition; in other words, it is not permitted to list some of the entries for a particular site in one part of a site sample file and the remainder of the entries in another part of the same file, with data pertaining to one or more other sites in between.
- A time entry of 24:00:00 is not permitted; this must be represented as 00:00:00 on the following day.

The integrity of a site sample file can be checked using program SMPCHEK documented herein. If any errors are present in a particular file of this type, they will be reported to the screen.

Site Listing File

A site listing file simply provides a list of sites, one to a line. Part of such a file is illustrated below.

```
13500002A
13500002B
13500005A
13500006A
13500007A
13500008A
13500009A
13500012A
13500015A
13500017A
13500023A
13500032A
13500032B
13500032C
site_a
site_b
site_c
site_d
site_e
site_f
site_g
site_h
```

Extract from a site listing file.

Different site listing files are often used in conjunction with a single site sample file, thus providing a mechanism whereby a subset of the latter can be selected for a particular type of processing.

A site listing file may possess more than one data column. If it does, columns after the first are ignored.

Site identifiers are case insensitive, being translated internally to upper case by programs of the Surface Water Utilities. No site should be cited more than once in a site listing file.