

In cooperation with the U.S. Environmental Protection Agency

TSPROC – A general time-series processor to assist in model calibration and result summarization

By Stephen M. Westenbroek, John Doherty, John F. Walker, Victor A. Kelson, and Randy J. Hunt

Report Series XXXX–2011

U.S. Department of the Interior

U.S. Geological Survey

U.S. Department of the Interior

KEN SALAZAR, Secretary

U.S. Geological Survey

Marcia K. McNutt, Director

U.S. Geological Survey, Reston, Virginia 2011

For product and ordering information:  
World Wide Web: http://www.usgs.gov/pubprod  
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth,  
its natural and living resources, natural hazards, and the environment:  
World Wide Web: http://www.usgs.gov  
Telephone: 1-888-ASK-USGS

Suggested citation:  
Westenbroek, S.M., Doherty, J., Walker, J.F., Kelson, V.A., and Hunt, R.J., 2011, TSPROC – A general time-series processor to assist in model calibration and result summarization: Middleton, Wis., United States Geological Survey.

Any use of trade, product, or firm names is for descriptive purposes only and does not imply   
endorsement by the U.S. Government.

Although this report is in the public domain, permission must be secured from the individual   
copyright owners to reproduce any copyrighted material contained within this report.

# Preface

Performance of this computer program has been tested and verified for many test cases; however, future applications of the program could reveal errors that were not detected in the test cases. Users are requested to notify the U.S. Geological Survey (USGS) if errors are found in the documentation report or in the computer program.

Correspondence regarding the report or program should be sent to

USGS Wisconsin Water Science Center

8505 Research Way

Middleton, WI 53562–3581

Attention: Stephen M. Westenbroek

Email: [smwesten@usgs.gov](mailto:smwesten@usgs.gov)

Although the computer program has been used by the USGS, no warranty, expressed or implied, is made by the USGS or the United States Government as to the accuracy and func­tionality of the program and related program material. Distribution of this software and source code do not consti­tute any such warranty, and no responsibility is assumed by the USGS in connection therewith.

The TSPROC code and other model-related programs are available for download­ing from the USGS at the following world wide web address: *http://water.usgs.gov/software/ground\_water.html*.

# Acknowledgements

The code described here was originally authored by John Doherty of Watermark Numerical Computing. The many organizations that supported John during initial development of the TSPROC code are gratefully acknowledged; these organizations include the University of Idaho, 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).

The United States Environmental Protection Agency is acknowledged for their support in developing this current version of TSPROC; their support allowed for the addition of numerous new capabilities, including routines for calculating period statistics, hydrologic indices, automated peak flow detection, hydrograph separation, and a module to access TSPROC from within a Python script.

# Contents

Preface iii

Acknowledgements iv

Contents v

Abstract 1

Introduction 1

Parameter Optimization and TSPROC 2

TSPROC capabilities 2

Model Calibration using TSPROC 5

Generation of control and instruction files 7

Using TSPROC 9

The TSPROC Input File - Overview 12

The DATE\_FORMAT and CONTEXT Settings 16

Blocks within a TSPROC Input File 17

DIGITAL\_FILTER 19

Butterworth Filter 21

Baseflow Separation Filter 22

Clipping (baseflow separation only) 23

Settling Time 24

Reverse Filtering 24

ERASE\_ENTITY 24

EXCEEDENCE\_TIME 25

GET\_MUL\_SERIES\_GSFLOW\_GAGE 28

GET\_MUL\_SERIES\_SSF 30

GET\_MUL\_SERIES\_STATVAR 32

GET\_MUL\_SERIES\_PLOTGEN 33

GET\_SERIES\_WDM 35

HYDRO\_EVENTS 36

HYDROLOGIC\_INDICES 37

LIST\_OUTPUT 39

NEW\_SERIES\_UNIFORM 41

NEW\_TIME\_BASE 43

PERIOD\_STATISTICS 44

REDUCE\_TIME\_SPAN 46

SERIES\_BASE\_LEVEL 47

SERIES\_CLEAN 48

SERIES\_COMPARE 50

SERIES\_DIFFERENCE 53

SERIES\_DISPLACE 54

SERIES\_EQUATION 56

SERIES\_STATISTICS 59

SETTINGS 61

USGS\_HYSEP 62

V\_TABLE\_TO\_SERIES 63

VOLUME\_CALCULATION 65

WRITE\_PEST\_FILES 67

Position within a TSPROC Input File 67

Model and Observation Entities 68

Keywords 68

Tasks Undertaken by TSPROC in Generating a PEST Input Dataset 69

Parameter and Parameter Group Data 71

Time Series Observations 73

S\_Table Observations 75

V\_Table Observations 76

G\_Table and E\_Table Observations 77

C\_Table Observations 77

The PEST Control File 77

Calibration using Patterns 79

References 80

Appendix 1: Site Sample File 82

Appendix 2: Python module use 84

Figures

**Figure 1.** TSPROC as part of a composite model undergoing parameter optimization with PEST. 5

**Figure 2.** TSPROC and PAR2PAR as part of a composite model undergoing optimization with PEST. 7

**Figure 3.** Invoking TSPROC - specifying control and run record filenames at the command prompt. 9

**Figure 4.** Invoking TSPROC - specifying control, run record filenames and context override at the command prompt. 9

**Figure 5.** Contents of a text file containing the responses to TSPROC prompts. 10

**Figure 6.** Example batch file snippets for starting a TSPROC run. 10

**Figure 7.** Command-line redirection prevents screen output from being written. 11

**Figure 8.** Example of a TSPROC control file. 12

**Figure 9.** Example usage of a DIGITAL\_FILTER block for use in applying a Butterworth filter. 19

**Figure 10.** Example usage of a DIGITAL\_FILTER block for use in applying a baseflow\_separation filter. 20

**Figure 11.** Example usage of a ERASE\_ENTITY block. 25

**Figure 12.** Example usage of a EXCEEDENCE\_TIME block. 25

**Figure 13.** Example of an EXCEEDENCE\_TIME block showing the use of the DELAY keyword. 27

**Figure 14.** Example usage of a GET\_MUL\_SERIES\_GSFLOW\_GAGE block. 29

**Figure 15.** Example usage of a GET\_MUL\_SERIES\_SSF block. 31

**Figure 16.** Example of the header and first three lines of a STATVAR file. 32

**Figure 17.** Example usage of a GET\_MUL\_SERIES\_STATVAR block. 33

**Figure 18.** Example usage of a GET\_MUL\_SERIES\_PLOTGEN block. 34

**Figure 19.** Example usage of a GET\_SERIES\_WDM block. 35

**Figure 20.** Example usage of a HYDRO\_EVENTS block. 36

**Figure 21.** Example usage of a HYDROLOGIC\_INDICES block. 39

**Figure 22.** Example usage of a LIST\_OUTPUT block. 40

**Figure 23.** Example usage of a NEW\_SERIES\_UNIFORM block 42

**Figure 24.** Example usage of a NEW\_TIME\_BASE block. 43

**Figure 25.** Example usage of a PERIOD\_STATISTICS block. 45

**Figure 26.** Example usage of a REDUCE\_TIME\_SPAN block. 47

**Figure 27.** Example usage of a SERIES\_BASE\_LEVEL block. 48

**Figure 28.** Example usage of a SERIES\_CLEAN block. 49

**Figure 29.** Example usage of a SERIES\_COMPARE block. 50

**Figure 30.** Example usage of a SERIES\_DIFFERENCE block. 54

**Figure 31.** Example usage of a SERIES\_DISPLACE block. 54

**Figure 32.** Example usage of a SERIES\_EQUATION block. 56

**Figure 33.** Examples of legal EQUATION arguments. 58

**Figure 34.** Example usage of a SERIES\_DIFFERENCE block. 60

**Figure 35.** Example usage of a SETTINGS block. 61

**Figure 36.** Example usage of a USGS\_HYSEP block. 63

**Figure 37.** Example usage of a SETTINGS block. 64

**Figure 38.** Example of a dates file. 65

**Figure 39.** Example usage of a VOLUME\_CALCULATION block. 66

**Figure 40.** Example of a WRITE\_PEST\_FILES block. 68

**Figure 41.** Example of a parameter data file. 72

**Figure 42.** Example of a parameter group file. 73

**Figure 43.** Example of valid weights equations. 74

**Figure 44.** Extract from a site sample file. 82

**Figure 45.** Example of simply starting TSPROC from within a Python script. 84

**Figure 46.** Example Python script that scans blocks in a TSPROC control file, processing the block that it has a routine for. 85

**Figure 47.** Example of Python script that creates TSPLOT blocks in-memory. 86

**Figure 48.** Example of Python script that processes all SSF files in the current directory. 87

Tables

**Table 1.** TSPROC input blocks. 17

**Table 2.** Keywords in a DIGITAL\_FILTER block. 19

**Table 3.** Keywords in an ERASE\_ENTITY Block. 25

**Table 4.** Keywords in an EXCEEDENCE\_TIME block. 25

**Table 5.** Keywords in a GET\_MUL\_SERIES\_GSFLOW\_GAGE block. 29

**Table 6.** Keywords in a GET\_MUL\_SERIES\_SSF block. 31

**Table 7.** Keywords in a GET\_MUL\_SERIES\_STATVAR block. 33

**Table 8.** Keywords in a GET\_MUL\_SERIES\_PLOTGEN block. 34

**Table 9.** Keywords within a GET\_SERIES\_WDM block. 35

**Table 10.** Keywords in a HYDRO\_EVENTS block. 36

**Table 11.** Keywords in a HYDROLOGIC\_INDICES block. 37

**Table 12.** Argument values for the STREAM\_CLASSIFICATION and FLOW\_COMPONENT keywords. 38

**Table 13.** Matrix of possible hydrologic indices selected by combining FLOW\_COMPONENT and STREAM\_CLASSIFICATION arguments. 38

**Table 14.** Keywords in a LIST\_OUTPUT block. 40

**Table 15.** Keywords in a NEW\_SERIES\_UNIFORM block. 42

**Table 16.** Keywords in a NEW\_TIME\_BASE block. 43

**Table 17.** Keywords in a PERIOD\_STATISTICS block. 45

**Table 18.** Keywords in a REDUCE\_TIME\_SPAN block. 47

**Table 19.** Keywords in a SERIES\_BASE\_LEVEL block. 48

**Table 20.** Keywords within a SERIES\_CLEAN block. 49

**Table 21.** Keywords within a SERIES\_COMPARE block. 50

**Table 22.** Keywords within a SERIES\_DIFFERENCE block. 54

**Table 23.** Keywords within a SERIES\_DISPLACE block. 54

**Table 24.** Keywords within a SERIES\_EQUATION block. 56

**Table 25.** Keywords in a SERIES\_STATISTICS block. 60

**Table 26.** Keywords within a SETTINGS block. 61

**Table 27.** Keywords within a USGS\_HYSEP block. 63

**Table 28.** Keywords in a V\_TABLE\_TO\_SERIES block. 64

**Table 29.** Keywords in a VOLUME\_CALCULATION block. 66

**Table 30.** Keywords in a WRITE\_PEST\_FILES block. 68

**Table 31.** Basic Python module methods. 84

Conversion Factors

Inch/Pound to SI

|  |  |  |
| --- | --- | --- |
| Multiply | By | To obtain |
| Length | | |
| inch (in.) | 2.54 | centimeter (cm) |
| foot (ft) | 0.3048 | meter (m) |
| mile (mi) | 1.609 | kilometer (km) |
| Area | | |
| acre | 0.4047 | hectare (ha) |
| acre | 0.004047 | square kilometer (km2) |
| square mile (mi2) | 2.590 | square kilometer (km2) |
| Volume | | |
| cubic foot (ft3) | 0.02832 | cubic meter (m3) |

Temperature in degrees Celsius (°C) may be converted to degrees Fahrenheit (°F) as follows:

°F=(1.8×°C)+32

Temperature in degrees Fahrenheit (°F) may be converted to degrees Celsius (°C) as follows:

°C=(°F-32)/1.8

SI to Inch/Pound

|  |  |  |
| --- | --- | --- |
| Multiply | By | To obtain |
| Length | | |
| centimeter (cm) | 0.3937 | inch (in.) |
| meter (m) | 3.281 | foot (ft) |
| kilometer (km) | 0.6214 | mile (mi) |
| Area | | |
| square meter (m2) | 0.0002471 | acre |
| square kilometer (km2) | 247.1 | Acre |
| square kilometer (km2) | 0.3861 | square mile (mi2) |
| Volume | | |
| cubic meter (m3) | 35.31 | cubic foot (ft3) |
| Flow rate | | |
| cubic meter per second (m3/s) | 35.31 | cubic foot per second (ft3/s) |

Temperature in degrees Celsius (°C) may be converted to degrees Fahrenheit (°F) as follows:

°F=(1.8×°C)+32

Temperature in degrees Fahrenheit (°F) may be converted to degrees Celsius (°C) as follows:

°C=(°F-32)/1.8

**TSPROC – A general time-series processor to assist in model calibration and result summarization**

By Stephen Westenbroek, John Doherty[[1]](#footnote-1), John Walker, Victor Kelson[[2]](#footnote-2), and Randy Hunt

# Abstract

The TSPROC (Time Series PROCessor) computer code has been developed in order to assist in the calibration of environmental models. The code is designed to perform a variety of calculations on time series data commonly associated with surface water models, including calculation of flow volumes, basic arithmetic processing, and generation of seasonal and annual statistics and hydrologic indices. TSPROC can also be used to generate some of the key input files required to perform parameter optimization by means of the PEST computer program. Through the use of TSPROC, the objective function for use in the model calibration process can be focused on specific components of a hydrograph.

# Introduction

TSPROC is a program designed to assist in the calibration of models, especially surface water models, by means of the widely used parameter estimation program PEST (Doherty 2009). TSPROC may also be used to create the basic input files required to make use of PEST.

Surface water models often simulate daily (or more frequent) discharge values at many locations and for many years. Unfortunately the thousands of data points contained in each hydrograph are serially related, and in the case of surface water models, more of this type of data is not necessarily better. Although parameter estimation may be successfully applied through use of daily hydrograph data, the resulting model calibration will not necessarily be able to replicate the aspects of the hydrograph pertinent to the project at hand. In addition, because of the amount of data involved (thousands or tens of thousands of records), automation of model post-processing tasks and PEST input file preparation is desirable.

This report documents the TSPROC code, and provides detail regarding each of the program options. Appendix A describes the format of a "site sample file", a generic ASCII file understood by TSPROC that may be used for storage, import, and export of time series data. Appendix B describes the development and basic use of a Python module version of TSPROC, which changes the way in which TSPROC may be used and expanded in the future.

# Parameter Optimization and TSPROC

This section introduces some basic definitions for terms used throughout the document, discusses general TSPROC capabilities, and describes how TSPROC may be used as part of a composite model subject to parameter optimization.

## TSPROC capabilities

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.

Many of the operations performed by TSPROC are designed specifically for use in the model calibration context. In order to make valid comparisons between the modeled and observed time series, the model-generated time series must be interpolated to the times at which observations were made. Because observations 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.

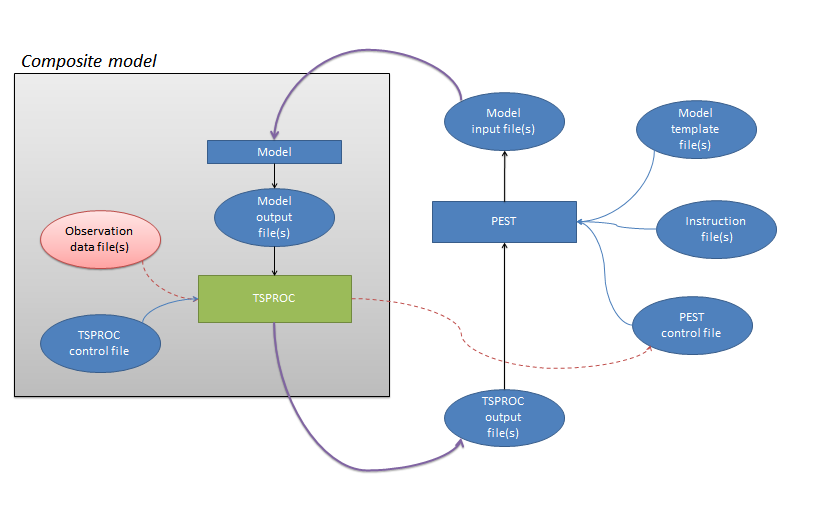
By using TSPROC it is possible to incorporate some or all of the following data types into the model calibration process:

1. **Raw data.** All or part of complete observation time series can be included; TSPROC can facilitate this by interpolating the model-generated series to field observation times.
2. **Processed data.** Time series data can be decomposed or filtered using the digital filtering or hydrograph separation techniques. 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). The current version of TSPROC also incorporates USGS HYSEP (HYdrograph SEParation) modules which include three different techniques for computing baseflow separation . Modeled and observed filtered counterparts can be individually matched through the calibration process.
3. **Accumulated volumes and masses.** 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 observations on the one hand and model-generated flows and constituent concentrations (interpolated to field observation 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 modeling applications it is crucial that a model predict exceedence-time characteristics as accurately as possible under future climatic/management conditions.
5. **Summary statistics** and **period statistics.** Basic statistics (mean, sum, median, maximum, minimum, range and standard deviation) can be calculated from the terms of a time series or functions of these terms over varying time intervals; the period statistics module allows for these statistics to be calculated on a monthly or annual basis over the length of the time series (for example, mean monthly), or as a series composed of monthly values (for example, monthly mean).
6. **Functions of arbitrary complexity** and **data patterns.** New time series may be calculated on the basis of one or more measured or modeled 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. 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 observation 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 observations into the parameter estimation process; see, for example, (Kuczera 1983). Relationships such as those used by the USGS LOADEST program (Runkel, Crawford, and Cohn 2004) may be suitable in some cases.
7. **Hydrologic indices.** When applied to surface water models, hydrologic indices may also be used to ensure that the calibration process results in a calibration faithful to particular aspects of the streamflow record. For example, if a model is being built to address the period and severity of low flow spells under a variety of landuse and climate scenarios, it is important to include indices capturing low flow characteristics in the model calibration process.

## Model Calibration using TSPROC

While TSPROC can be run as an independent executable program, TSPROC is most useful when run as part of a composite model by PEST. A composite model is a model made up of two or more executable programs run in succession by means of 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 more of the time series generated by the model. Similar operations can be carried out on time series comprised of observation data. The processed observations 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 (Figure 1).

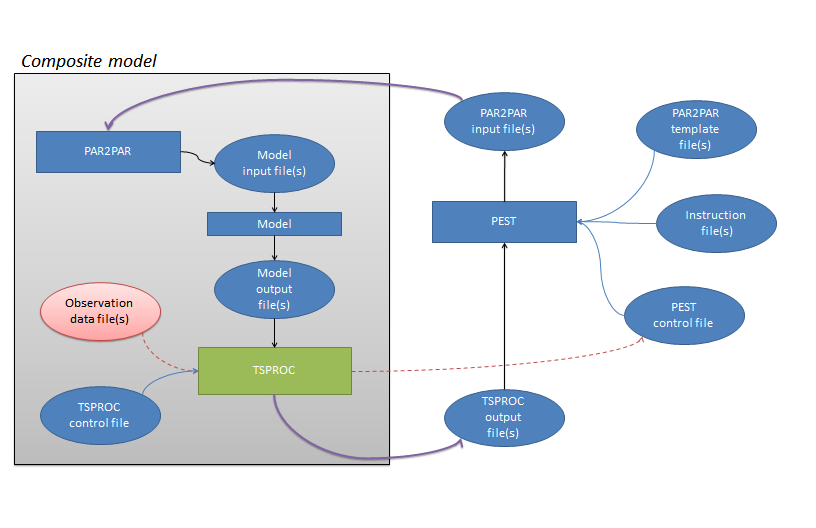
1. TSPROC as part of a composite model undergoing parameter optimization with PEST.



Upon startup of a PEST optimization run, PEST modifies one or more model parameters on the basis of information gleaned from the model response relative to parameter changes (Jacobian matrix). Updated model parameters are written to a new model input file, using the model template file as a map to locate positions at which the new parameter values are to be substituted. The composite model (a batch or shell script) is then run, which in turn runs the model code, followed by TSPROC. TSPROC performs the input, processing, and output tasks as specified in the TSPROC control file, generating a composite model output file (TSPROC output file). PEST reads this new output file, with assistance from the instruction file, calculates a new set of model parameters to try, and the optimization cycle continues.

The composite model may contain multiple processing steps, calculated by many different executable codes. A preprocessor, such as the PAR2PAR code, is often added to the composite model in order to distribute PEST-estimated parameters to a large number of model computational elements (Figure 2).

1. TSPROC and PAR2PAR as part of a composite model undergoing optimization with PEST.



PAR2PAR may be used, for example, if the modeler wishes to distribute a generic parameter estimated by PEST to a large number of hydrologic response units (HRU) in a surface water model on the basis of, say, the predominant glacial geology of each HRU. PEST writes a PAR2PAR input file on the basis of the user-provided PAR2PAR template file. In turn, PAR2PAR transforms the single generic parameter value into individual values for each HRU, incorporating them into a model input file.

## Generation of control and instruction files

As can be seen from the preceding description, TSPROC is a program of considerable complexity. TSPROC can be used to generate the PEST control file and the instruction file needed to interpret the composite model output (Figure 1, Figure 2). Raw and processed observation values are embedded in the PEST control file, and should not need to be recalculated during model optimization; it thus makes sense to use the TSPROC context mechanism to keep operations involving observed values separate from those associated with simulated values. A “pest\_prep” context might be used to perform input, processing, and output on observed values, while an “optimize” context might be used during the actual PEST runs to process only the simulated values and not the observed values. If one is processing a file with a large number (greater than 10,000 lines of control file input) of records, the time saved by not recalculating statistics on the observed values with each iteration can be susbstantial.

A typical application of TSPROC within a model calibration / optimization process includes the following tasks:

1. Process various types of observation data to generate an appropriate observation dataset. This task includes reading the observation data from a file, an optional cleaning step to remove questionable or missing data, and as many steps as needed to produce the volumes, statistics and indices desired in the calibration process.
2. Time-interpolate raw model-generated time series to the times at which field observations were made, then process that data in an identical fashion to that in which the observation data were processed.
3. Generate a PEST input data files. These include a PEST control file recording the observations used in the calibration process, and an instruction file capable of reading the model-generated counterparts to these observed quantities from the appropriate model output file.

In order to generate PEST input data files, TSPROC must be provided with the set of template files pertinent to the current calibration exercise; template files are simply a set of normal model input files, except that the names of the parameters to be adjusted are substituted in place of numerical values (for example, “% Kv\_coef %” in place of “3.15”). TSPROC will write a complete PEST control file in which these parameters are recorded, together with the processed observations 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 observation type. TSPROC will then write the instruction file informing PEST how to read the model-generated equivalents to these processed observations from a TSPROC output file when the latter is run as part of a composite model by PEST.

## Using TSPROC

TSPROC is executed from the command line; three optional command line arguments may be supplied. If no arguments are supplied, TSPROC will prompt the user for the name of the TSPROC control file and the name of the output run record file. Figure 1 demonstrates the syntax used when specifying the TSPROC control filename and the run record filename from the command line. Figure 2 demonstrates how the context (discussed below) for the TSPROC run may be overridden when specifying the command and run record filenames.

1. Invoking TSPROC - specifying control and run record filenames at the command prompt.

tsproc *control\_filename run\_record\_filename*

1. Invoking TSPROC - specifying control, run record filenames and context override at the command prompt.

tsproc *control\_filename run\_record\_filename* CONTEXT *context\_name*

The TSPROC control file specifies file inputs and outputs as well as the type of time series processing to be performed on each. As TSPROC runs, it echoes the contents of its input file and the operations that it performs to both the screen and to the 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.

If TSPROC is run by PEST as part of a composite model, it is convenient to supply the control and run record filenames as part of the command syntax used to start a TSPROC run as demonstrated in Figure 1.

Previous versions of TSPROC did not accept command-line arguments, requiring the responses to TSPROC’s prompts to be placed into a text file prior to the PEST run and provided to TSPROC through the command-line redirection mechanism. The current version of TSPROC may still be invoked in this manner. For example, to invoke TSPROC given an input file named tsproc.dat and a run record file named tsproc.rec, then a text file (named, for example, tsproc.in) could be prepared as shown in Figure 3.

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

tsproc.dat

tsproc.rec

When TSPROC is then run as part of a composite model by PEST, the composite model batch file should invoke TSPROC in one of the two ways shown in Figure 6:

1. Example batch file snippets for starting a TSPROC run.

:: The suggested syntax for starting TSPROC from with a batch

:: file is shown below:

tsproc tsproc.dat tsproc.rec

:: The traditional way to start TSPROC from within a batch file is:

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.

Any errors encountered while running TSPROC will cause a report to be written to both the screen and to the run record file. Once an error is detected, TSPROC will not read the 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). 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 syntax as shown in

1. Command-line redirection prevents screen output from being written.

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 give 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. Figure 6 gives a short example showing the contents of a TSPROC control file.

1. Example of a TSPROC control file.

####################################################################

# The ‘SETTINGS’ block must be first within the control file.

####################################################################

START SETTINGS

CONTEXT pest\_input

DATE\_FORMAT mm/dd/yyyy

END SETTINGS

####################################################################

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

####################################################################

START GET\_MUL\_SERIES\_PLOTGEN

CONTEXT all

FILE catchment.plt

LABEL "total outflow"

NEW\_SERIES\_NAME flow\_mod

END GET\_MUL\_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

####################################################################

# Modeled 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 modeled 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 modeled 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

####################################################################

# Modeled 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

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 the exception of the settings block, blocks may 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 the SETTINGS block must come before all other blocks. The SETTINGS block must contain two keywords: DATE\_FORMAT and CONTEXT.

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: 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.

If at least one of the CONTEXT strings supplied in a processing block is "all", 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

Table 1 lists the blocks which may be present within any TSPROC input file. Multiple occurrences of any block except the SETTINGS block are permitted.

1. TSPROC input blocks.

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 18 characters or less in length.

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 18 characters or less in length, this name being provided by the user following the NEW\_C\_TABLE, NEW\_S\_TABLE, NEW\_V\_TABLE, NEW\_G\_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; 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.

The remainder of this report section contains detailed discussions of the TSPROC blocks, arranged in alphabetical order. Each block must contain one or more CONTEXT keywords and arguments as the first entries in the block; other keyword entries may appear in any order within a block, unless noted otherwise.

## DIGITAL\_FILTER

The DIGITAL\_FILTER block instructs TSPROC to calculate a new time series by passing an existing series 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, and which has enough terms for the filtering algorithm to work. Thus before performing filtering operations TSPROC checks the specified 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.) TSPROC will not allow filtering operations to take place on a time series that has fewer than 20 entries.

Keywords available in the DIGITAL\_FILTER block are listed inTable 2. Example application of a Butterworth filter within a DIGITAL\_FILTER block is shown in Figure 9; application of a baseflow separation filter is shown in Figure 10.

1. Keywords in a DIGITAL\_FILTER block.
2. Example usage of a DIGITAL\_FILTER block for use in applying a Butterworth filter.

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

1. Example usage of a DIGITAL\_FILTER block for use in applying a baseflow\_separation filter.

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

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. TSPROC 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.

Use of each of the types of digital filter implemented by TSPROC is discussed in the following sections.

### 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 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; note than 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. 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.

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 decrease 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 for most applications in surface water hydrology.

### 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 (baseflow separation only)

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

A digital 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.

### 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 or 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 Table 3; an example of an ERASE\_ENTITY block is shown in Figure 11.

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

1. Keywords in an ERASE\_ENTITY Block.
2. Example usage of a 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

## 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 Table 4; an example of an EXCEEDENCE\_TIME block is shown in Figure 12. 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.

1. Keywords in an EXCEEDENCE\_TIME block.
2. Example usage of a 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

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 in Figure 13.

1. Example of an EXCEEDENCE\_TIME block showing the use of 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 Table 5; an example of a GET\_MUL\_SERIES\_GSFLOW\_GAGE block is shown in Figure 14. Keywords may be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

1. Keywords in a GET\_MUL\_SERIES\_GSFLOW\_GAGE block.
2. Example usage of 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

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.

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 GET\_MUL\_SERIES\_SSF block allows one or more series to be imported from a single site sample file in one operation. 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. Note that for compatibility with previous versions of TSPROC, a “GET\_SERIES\_SSF” block will be processed as described in this section as well.

Table 6 below shows the keywords appearing in a GET\_MUL\_SERIES\_SSF block. An example of a GET\_MUL\_SERIES\_SSF block is shown in Figure 15.

1. Keywords in a GET\_MUL\_SERIES\_SSF block.
2. Example usage of a GET\_MUL\_SERIES\_SSF block.

START GET\_MUL\_SERIES\_SSF

CONTEXT all

FILE flows.ssf

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

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. These keywords must be supplied in pairs, with the SITE keyword immediately preceding the associated NEW\_SERIES\_NAME keyword. 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.

## GET\_MUL\_SERIES\_STATVAR

STATVAR files are written by the USGS Precipitation-Runoff Modeling System (PRMS) model, as well as by the USGS GSFLOW model. An example of such a file is given in Figure 16.

1. Example of the header and first three lines of a STATVAR file.

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

*{…continues…}*

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, which generally corresponds to a particular hydrologic response unit (HRU), Muskingum stream node, or other reservoir. 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 Table 7. An example of a GET\_MUL\_SERIES\_STATVAR block is given in Figure 17.

1. Keywords in a GET\_MUL\_SERIES\_STATVAR block.
2. Example usage of 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 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\_MUL\_SERIES\_PLOTGEN

The GET\_MUL\_SERIES\_PLOTGEN block governs importation of time series data from a HSPF PLOTGEN file into TSPROC. 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. Table 8 lists the allowable keywords in a GET\_MUL\_SERIES\_PLOTGEN block. Note that for compatibility with previous versions of TSPROC, a control file that contains a “GET\_SERIES\_PLOTGEN” block will be recognized and treated as a GET\_MUL\_SERIES\_PLOTGEN block as well. Figure 18 shows an example of a GET\_MUL\_SERIES\_PLOTGEN block.

1. Keywords in a GET\_MUL\_SERIES\_PLOTGEN block.
2. Example usage of a GET\_MUL\_SERIES\_PLOTGEN block.

START GET\_MUL\_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\_MUL\_SERIES\_PLOTGEN

DATE\_ and TIME\_ specifiers are optional in a GET\_MUL\_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\_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.

Table 9 below shows the keywords permissible in a GET\_SERIES\_WDM block. An example of a GET\_SERIES\_WDM block is given in Figure 19.

1. Keywords within a GET\_SERIES\_WDM block.
2. Example usage of 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

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 behavior 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.

## HYDRO\_EVENTS

The HYDRO\_EVENTS block allows storm hydrographs to be extracted for the days preceding and following peak flow events; events may defined by specifying a minimum number of days between peaks and by assigning a minimum peak value. The terms of the series upon which events are extracted may be limited to those events occurring within a specified date/time interval.

TSPROC stores the outcomes of storm events extracted by the HYDRO\_EVENTS block in a new time series. Like other TSPROC entities, the new time series must be provided with a name so that it can be referenced by other TSPROC processing blocks. This name must be 18 characters or less in length and must not include a space character.

Keywords featured in the HYDRO\_EVENTS block are listed in Table 10. An example of a HYDRO\_EVENTS block is given in Figure 20.

1. Keywords in a HYDRO\_EVENTS block.
2. Example usage of a HYDRO\_EVENTS block.

START HYDRO\_EVENTS

CONTEXT all

SERIES\_NAME outflow

NEW\_SERIES\_NAME outflow\_pk

WINDOW 7

MIN\_PEAK 10.5

RISE\_LAG 2

FALL\_LAG 6

DATE\_1 10/1/1976

TIME\_1 00:00:00

DATE\_2 9/30/1985

TIME\_2 00:00:00

END HYDRO\_EVENTS

## HYDROLOGIC\_INDICES

Hydrologic indices are statistical measures applied to streamflow records in an attempt to quantify various ecologically important aspects of a flow regime. Classes of hydrologic indices include measures that quantify such things as the timing and seasonal pattern of extreme flows, daily, seasonal, and annual flow variability, as well as rates of flow increases and decreases. Over 160 different hydrologic indices may be calculated by means of a HYDROLOGIC\_INDICES block. Computation of the indices is accomplished by means of code adapted from the USGS Hydrologic Index Tool (Henricksen et al. 2006).

The list of hydrologic indices to be calculated may be specified in one of two ways: 1) by means of the STREAM\_CLASSIFICATION and FLOW\_COMPONENT keywords, or 2) through use of the two-letter keywords (for example, “MA”), followed by an index number.

Table 11 lists the keywords available for use in a HYDROLOGIC\_INDICES block. Note that all indices will be calculated and reported if the user specifies no hydrologic indices nor supplies arguments to the STREAM\_CLASSIFICATION or FLOW\_COMPONENT keywords.

1. Keywords in a HYDROLOGIC\_INDICES block.

Note also that the two-letter keywords will accept any number of values (space-delimited) as arguments. For example, the following keyword and arguments could be used to specify that the indices associated with magnitude of average flows, numbers 12, 22, and 25 are calculated: “MA 12 22 25”.

The possible argument values that may be specified for the STREAM\_CLASSIFICATION and FLOW\_COMPONENT keywords are shown in Table 12.

1. Argument values for the STREAM\_CLASSIFICATION and FLOW\_COMPONENT keywords.

Because many of the hydrologic indices that can be calculated are somewhat redundant, the user may elect to calculate a subset of the available indices on the basis of the stream’s hydrologic regime type (for example, snowmelt-dominated, groundwater-dominated, surface runoff-dominated). Olden and Poff identified a list of the high-information, non-redundant indices for six stream classification types. Table 13 summarizes the arguments that may be supplied to the STREAM\_CLASSIFICATION and FLOW\_COMPONENT keywords in order to generate a subset of hydrologic indices appropriate to the system under study. The hydrologic index definitions referenced in Table 13 are those given in Olden and Poff (2003).

1. Matrix of possible hydrologic indices selected by combining FLOW\_COMPONENT and STREAM\_CLASSIFICATION arguments.

For example, specifying “STREAM\_CLASSIFICATION snowmelt\_perennial” and “FLOW\_COMPONENT high\_flow\_frequency” would result in the calculation and reporting of just two indices: FH8 and FH11. Subsequent FLOW\_COMPONENT entries will append the appropriate indices to the list to be calculated. Multiple instances of both the STREAM\_CLASSIFICATION and the FLOW\_COMPONENT keywords may be present in the control file. Only the most recent argument value is retained for the STREAM\_CLASSIFICATION keyword. If only the STREAM\_CLASSIFICATION or FLOW\_COMPONENT keyword is given (not both), this has the effect of selecting all indices identified by Olden and Poff associated with the given stream classification or flow component. In other words, specifying “STREAM\_CLASSIFICATION flashy\_intermittent” will result in calculation of the indices associated with a flashy intermittent stream type for all flow components.

Figure 21 shows an example HYDROLOGIC\_INDICES block.

1. Example usage of a HYDROLOGIC\_INDICES block.

START HYDROLOGIC\_INDICES

CONTEXT all

SERIES\_NAME outflow

NEW\_TABLE\_NAME outflow\_hi

STREAM\_CLASSIFICATION groundwater\_perennial

FLOW\_COMPONENT timing

FLOW\_COMPONENT low\_flow\_magnitude

MA 3 4 10

DATE\_1 01/01/1980

TIME\_1 00:00:00

DATE\_2 12/31/1989

TIME\_2 23:59:59

END HYDROLOGIC\_INDICES

The block shown in Figure 21 would cause TSPROC to calculate the following hydrologic indices: ML14, ML16, ML18, MA3, MA4, MA10, TA1, TH1, TL2. The calculated results are added to a new G\_TABLE (general table), which may be manipulated or output just like any other TSPROC table.

## LIST\_OUTPUT

The LIST\_OUTPUT block causes the series and tables generated by TSPROC to be written to an ASCII text list output file or site sample file. 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 given in Table 13. An example of a LIST\_OUTPUT block is shown in Figure 22. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

1. Keywords in a LIST\_OUTPUT block.
2. Example usage of 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

Any number of time series or tables can be written to a file generated by the LIST\_OUTPUT block; as many of the keywords regarding 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, followed by e\_tables, and finally by g\_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", "long", and “ssf”. If “short” is specified, the LIST\_OUTPUT block will list the terms of the time series as a single column in its output file. If the “long” option is specified 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 SERIES\_FORMAT is specified as “ssf”, TSPROC will not write a header for each time series, and the user is spared having to remove the header of a list output file manually.

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, g\_tables and c\_tables, it is important to note that statistics not requested when the block is created are not recorded in the file written by the LIST\_OUTPUT block.

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 Table 14. An example NEW\_SERIES\_UNIFORM block is given in Figure 23.

1. Keywords in a NEW\_SERIES\_UNIFORM block.
2. Example usage of 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

The behavior 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 user-supplied NEW\_SERIES\_VALUE.

The NEW\_TIME\_SERIES block can be useful as a precursor to digital filtering; 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 values present at an irregular time increment, the series can be interpolated to a constant time base series by:

1. creating a series with uniform time increments by means of a NEW\_SERIES\_UNIFORM block; and
2. creating a new series (with regular time intervals and interpolated values) by means of the NEW\_TIME\_BASE block, specifying the series created in step 1 above as the new time base.

Filtering can then be undertaken on the interpolated time series.

## NEW\_TIME\_BASE

The NEW\_TIME\_BASE block is used to interpolate values 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 Table 15. An example NEW\_TIME\_BASE block may be found in Figure 24.

1. Keywords in a NEW\_TIME\_BASE block.
2. Example usage of 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

Time interpolation of one time series to the time-base of another will only occur if the time-span of the series specified by TB\_SERIES\_NAME is equal to, or smaller than, that of the series (specified by SERIES\_NAME) to be interpolated. The result will be a new time series with terms that have 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.

## PERIOD\_STATISTICS

Using the PERIOD\_STATISTICS block a number of simple statistics can be calculated for specific periods within 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 PERIOD\_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, 7 statistical measures can be calculated using the PERIOD\_STATISTICS block: these include the mean, standard deviation, median, sum, maximum, minimum and range. The statistics may be calculated for a monthly series (for example, “12/1997”, “01/1998”, “02/1998”), or monthly.TSPROC stores the outcomes of statistical calculations carried out by the PERIOD\_STATISTICS block in a new time series. Like other TSPROC entities, the new time series must be provided with a name so that it can be referenced by other TSPROC processing blocks. This name must be 18 characters or less in length and must not include a space character.

Keywords featured in the PERIOD\_STATISTICS block are listed in Table 16. An example of a PERIOD\_STATISTICS block is shown in Figure 25.

1. Keywords in a PERIOD\_STATISTICS block.
2. Example usage of a PERIOD\_STATISTICS block.

START PERIOD\_STATISTICS

CONTEXT all

SERIES\_NAME outflow

NEW\_SERIES\_NAME outflow\_m

STATISTIC mean

PERIOD month\_many

TIME\_ABSCISSA start

LOG yes

DATE\_1 10/1/1976

TIME\_1 00:00:00

DATE\_2 9/30/1985

TIME\_2 00:00:00

END PERIOD\_STATISTICS

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; 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 high-flow water year begins October 1 of the previous year and ends September 30 of an individual year. The low-flow water year begins April 1 of the previous year and ends March 31 of an individual year. The calendar year begins January 1 and ends December 31 of an individual year.

For monthly statistics, selecting “period\_many” as the PERIOD keyword will compute a monthly statistic for each year in the input time series. Alternatively, selecting “period\_one” as the PERIOD keyword will compute a single statistic for each month, aggregating across all years in the input time series. When computing a mean, the “period\_many” mean is commonly called the monthly mean value, whereas the “period\_one” mean is commonly called the mean monthly value.

## 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 observations were made, to be carried out using the NEW\_TIME\_BASE block.

Keywords found in a REDUCE\_TIME\_SPAN block are listed in Table 17. An example of a REDUCE\_TIME\_SPAN block is shown in Figure 26.

1. Keywords in a REDUCE\_TIME\_SPAN block.
2. Example usage of 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

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; in other words, 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. 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 Table 18. An example of a SERIES\_BASE\_LEVEL block is given in Figure 27.

1. Keywords in a SERIES\_BASE\_LEVEL block.
2. Example usage of a SERIES\_BASE\_LEVEL block.

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 reverses the sign of the output time series; this can be useful, for example, when calculating drawdown from a time series of head observations. 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 misbehavior 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 Table 19. An example SERIES\_CLEAN block is shown in Figure 28.

1. Keywords within a SERIES\_CLEAN block.
2. Example usage of 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

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 1.0E+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 Table 20. An example SERIES\_COMPARE block is given in Figure 29.

1. Keywords within a SERIES\_COMPARE block.
2. Example usage of 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

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 (Nash and Sutcliffe 1970), 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 given below. The symbols Si and Oi refer to individual values in a simulated and observed series, respectively. 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:

 (1)

Standard error:

 (2)

Relative bias:

 (3)

Relative standard error:

 (4)

Nash-Sutcliffe coefficient:

 (5)

Coefficient of efficiency:

` (6)

*Index of agreement:*

 (7)

where:

 (8)

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, in the equations for coefficient of efficiency and index of agreement is replace by Bi, 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 Table 21. An example SERIES\_DIFFERENCE block is given in Figure 30.

1. Keywords within a SERIES\_DIFFERENCE block.
2. Example usage of a SERIES\_DIFFERENCE block.

A SERIES\_DIFFERENCE block.

START SERIES\_DIFFERENCE

CONTEXT all

SERIES\_NAME 322

NEW\_SERIES\_NAME 322\_d

END SERIES\_DIFFERENCE

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. The new time series has one less term than the original time series; the date and time attributed to the first value of the new series is that of the second value present in the original series. For example, if a series spans the period 1/1/1990 to 2/1/1990, the value of the first term of the new series represents the January 2nd value minus the January 1st value; the date associated with this difference would be, in this case, January 2, 1990.

## 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 Table 22. An example SERIES\_DISPLACE block is given in Figure 31. Keywords can be supplied in any order, except for the CONTEXT keyword(s) which must precede all others.

1. Keywords within a SERIES\_DISPLACE block.
2. Example usage of 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

The SERIES\_DISPLACE operation requires that the time series upon which the operation is carried out has 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 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 (differences between simulated and observed values) 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 inherent 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 Table 23. An example of a SERIES\_EQUATION block is shown in Figure 32.

1. Keywords within a SERIES\_EQUATION block.
2. Example usage of a SERIES\_EQUATION block.

START SERIES\_EQUATION

CONTEXT all

NEW\_SERIES\_NAME new\_series

EQUATION log10(outflow \* concentration)

END SERIES\_EQUATION

The terms of the new time series 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.

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; if in doubt, use parentheses to set precedence between operators.

An equation supplied in the SERIES\_EQUATION block can include most of the commonly-used mathematical functions: 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; and
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 defines two other specializd functions for use 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.

Figure 33 presents examples of some valid series equations.

1. Examples of legal EQUATION arguments.

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")

An equation may be as simple as the first of the equations shown in Figure 33. Although not terribly useful, applying this equation would result in a new series created using the unaltered terms of the existing series outflow.

In the fourth 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 fifth of the above equations the argument of the sine function is multiplied by  in order to achieve periodicity of one year.

For those unfamiliar 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.

Currently 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 if the user intends 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 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 18 characters or less in length and must not include a space character.

Keywords featured in the SERIES\_STATISTICS block are listed in Table 24. An example of a SERIES\_STATISTICS block is given in Figure 34.

1. Keywords in a SERIES\_STATISTICS block.
2. Example usage of a SERIES\_DIFFERENCE 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

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. 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. Two keywords must be used in a SETTINGS block; both of these are mandatory. Table 25 lists the allowable keywords within a SETTINGS block. Figure 35 shows an example SETTINGS block.

1. Keywords within a SETTINGS block.
2. Example usage of a SETTINGS block.

START SETTINGS

DATE\_FORMAT mm/dd/yyyy

CONTEXT pest\_input

END SETTINGS

The DATE\_FORMAT setting allows TSPROC to adapt to the different methods by which the date is represented in different countries. Currently TSPROC understands two date formats: mm/dd/yyyy and dd/mm/yyyy.

A SETTINGS block can contain only one CONTEXT keyword. Through use of various context arguments the user may activate or inactivate combinations of TSPROC blocks as needed. 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.

TSPROC blocks may be activated or inactivated through the choice of the argument given for the CONTEXT keyword. 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.

## USGS\_HYSEP

Using the USGS\_HYSEP block, baseflow can be separated from total flow for specific periods within a time series. The user may select one of three different baseflow separation methods for application within this block.

TSPROC stores the outcomes of baseflow\_separation by the USGS\_HYSEP block in a new time series. Like other TSPROC entities, the new time series must be provided with a name so that it can be referenced by other TSPROC processing blocks. This name must be 18 characters or less in length and must not include a space character.

Keywords featured in the USGS\_HYSEP block are listed in Table 26. An example of a USGS\_HYSEP block is given in Figure 36.

1. Keywords within a USGS\_HYSEP block.
2. Example usage of a USGS\_HYSEP block.

The techniques used for baseflow separation are described in the following publication: Sloto, R.A., and Crouse, M.Y., 1996, HYSEP: A computer program for streamflow hydrograph separation and analysis: U.S. Geological Survey Water-Resources Investigations Report 96-4040, 46 p.

Each technique uses a time interval to perform the separation, which is usually based on the following formula for time of concentration (time interval following cessation of rainfall where surface runoff occurs):



Where tc is time of concentration in days, and A is drainage area in mi2. In general, the interval used should be twice the width of the time of concentration, rounded to the nearest odd integer. However, any odd integer can be entered for the TIME\_INTERVAL keyword.

## 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 Table 27. An example of a V\_TABLE\_TO\_SERIES block is given in Figure 37.

1. Keywords in a V\_TABLE\_TO\_SERIES block.
2. Example usage of a SETTINGS block.

START V\_TABLE\_TO\_SERIES

CONTEXT all

V\_TABLE\_NAME volume

NEW\_SERIES\_NAME ssvol

TIME\_ABSCISSA end

END V\_TABLE\_TO\_SERIES

While there are two dates and times associated with every term of a v\_table (corresponding to the interval over which volume is accumulated), there is only one date and time associate with every time series entry. In the process of converting from a table to a series 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 centers 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 "center" (or "centre") 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 a dates file is shown in Figure 38. 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.

1. Example of a dates file.

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 can be of any length. Each line must contain 4 entries: 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 Table 28. An example of a VOLUME\_CALCULATION block is given in Figure 39.

1. Keywords in a VOLUME\_CALCULATION block.
2. Example usage of 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

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

The WRITE\_PEST\_FILES block instructs TSPROC to generate PEST input files for a parameter estimation run. It is assumed that the control file supplied to TSPROC is almost identical whether used as part of a composite model or in generating the PEST control files used in the parameter estimation process. The CONTEXT keyword in the SETTINGS block should be used to activate the WRITE\_PEST\_FILES block during PEST control file generation, and to deactivate this block when TSPROC is used as part of a composite model.

It is beyond the scope of this publication to discuss parameter estimation using PEST software on more than a superficial basis; readers should consult PEST documentation for more detail on the use and application of PEST.

### 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 and tables cited in the LIST\_OUTPUT block are generated by the model under calibration. 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.

### Model and Observation Entities

Model-generated and observed time series and tables must be comparable; they must contain the same number of values, and cover identical date and time ranges. Should TSPROC detect any inconsistencies in such paired entities, it will cease execution with an appropriate error message.

It is generally a good idea to time-interpolate model-generated series to the times and dates of the observation time series to which they correspond before performing any further analyses on them. 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.

### Keywords

Table 29 describes the keywords associated with a WRITE\_PEST\_FILES block.

1. Keywords in a WRITE\_PEST\_FILES block.

An example of a WRITE\_PEST\_FILES block is given in Figure 40. Note that the WRITE\_PEST\_FILES block may include an almost unlimited number of paired comparisons between smodel-generated and observed series and tables.

1. Example of a WRITE\_PEST\_FILES block.

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

### 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. 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 and stores the data for later use.
3. If a PARAMETER\_GROUP\_FILE keyword is present within the WRITE\_PEST\_FILES block, TSPROC reads that file and stores the data for later use.
4. TSPROC checks that all model series and 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. TSPROC checks that each observation entity that is matched to a model entity is comparable; in other words, it checks to ensure that table comparisons are being made between indentical table types, and that series contain identical numbers of values and cover identical time and date ranges.
6. TSPROC then generates names for all observations featured in the parameter estimation process.
7. TSPROC 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 accumulates 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. Each TEMPLATE\_FILE keyword should be followed by a MODEL\_INPUT\_FILE keyword. In so doing, PEST links the 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.

An example of a parameter data file is given in Figure 41.

1. Example of a parameter data file.

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

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 of a parameter group file is given in Figure 42.

1. Example of a parameter group file.

ro relative 0.01 0.00 switch 1.5 parabolic

h relative 0.01 1.0e-4 switch 2.0 parabolic

### 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 observation 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.

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 issue 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 are given in Figure 43.

1. Example of valid weights equations.

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 set to the values of an existing time series. 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 in the argument to the sine function). Recall that the argument to the sin, cos and tan functions must be supplied in radians; 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 generated by 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.

### G\_Table and E\_Table Observations

Inclusion of g\_table or 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

The PEST control file written by TSPROC will likely need to be edited by the user in order to ensure that it is compatible with the optimization problem at hand. 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; these values are suitable for most occasions. If it is desired that PEST run in regularisation mode, a set of regularisation observations and/or prior information equations must also be added by hand to the TSPROC-generated PEST control file.

If a MODEL\_COMMAND\_LINE keyword is provided in a WRITE\_PEST\_FILES block, the user-supplied command line argument 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 the composite model (in other words, the name of the batch or script file which runs the model or models along with PAR2PAR, TSPROC and any other glue code). Commands cited in this file will include the name of a model executable, as well as the command to run TSPROC.

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.

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, because only one 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.

Although 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.

### 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 observations (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 observations due to the erratic and noisy nature of these observations.

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 observations 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 observations 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 modeled constituent concentrations and those calculated from modeled 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 observation 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

Doherty, John. 2009. *PEST: Model-independent Parameter Estimation User Manual*. 5th ed. Brisbane, Australia: Watermark Numerical Computing.

Henricksen, J.A., J. Heasley, J.G. Kennen, and S. Nieswand. 2006. *Users’ manual for the Hydroecological Integrity Assessment Process software (including the New Jersey Assessment Tools)*. Open-File Report 2006-1093. U.S. Geological Survey. http://www.fort.usgs.gov/Products/Publications/pub\_abstract.asp?PubID=21598.

Kuczera, G. 1983. Improved parameter inference in catchment models 2. combining different kinds of hydrologic data and testing their compatibility. *Water Resources Research* 19, no. 5: 1163–1172.

Legates, D. R, and G. J McCabe. 1999. Evaluating the use of “goodness-of-fit” measures in hydrologic and hydroclimatic model validation. *Water Resources Research* 35, no. 1: 233–241.

Nash, J. E., and J. V. Sutcliffe. 1970. River flow forecasting through conceptual models part I–A discussion of principles. *Journal of hydrology* 10, no. 3: 282–290.

Nathan, R. J., and T. A. McMahon. 1990. Evaluation of automated techniques for base flow and recession analyses. *Water Resources Research* 26, no. 7: 1465–1473.

Olden, Julian D., and N. L. Poff. 2003. Redundancy and the choice of hydrologic indices for characterizing streamflow regimes. *River Research and Applications* 19, no. 2 (March): 101-121. doi:10.1002/rra.700.

Peterson, P. 2009. F2PY: a tool for connecting Fortran and Python programs. *International Journal of Computational Science and Engineering* 4, no. 4: 296–305.

Ronald, A., and M. Y.C Sloto. 1996. Hysep: A computer program for streamflow hydrograph separation and analysis. *US. Geological survey*.

Runkel, R.L., Crawford, C.G. and Cohn, T.A., 2004, Load Estimator (LOADEST): A FORTRAN program for estimating constituent loads in streams and rivers, Techniques and Methods book 4, chapter A5: US Geological Survey, Reston, VA.

# Appendix 1: 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.

1. Extract from a site sample file.

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

The first item on each line of a site sample file is a site identifier. This identifier must be of 18 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 observation 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.

# Appendix 2: Python module use

The current version of TSPROC can be compiled as a traditional Fortran program or as a Fortran library with bindings to Python. Many new ways of using and extending TSPROC are possible through the use of the Python module and Fortran library. This is a capability that will likely continue to evolve long after the publication of this document; users should check online documentation for the most current information regarding the capabilities of the Python TSPROC module.

The Fortran wrapping package f2py has been used to assist in creating entry points to the Fortran code from within Python. The TSPROC code may be used in a number of different ways from within a Python script. The simplest application invokes TSPROC from within a Python script as opposed to from a the command line. A mode complex Python script can create TSPROC input blocks in memory, obviating the need for a control file. Lastly, the TSPROC module may be used in such a way that the Python script simply monitors the block currently being processed; this allows the user the option to override or extend current TSPROC functionality . The basic Python module methods are listed in Table 31.

1. Basic Python module methods.

Figure 45 shows a Python script that simply initializes the TSPROC library and then starts a TSPROC run with no further processing from the Python script. This could be of value if Python were used as the scripting language to make up the composite model rather than a batch file or shell script.

1. Example of simply starting TSPROC from within a Python script.

import pytsproc.main\_loop as tsp

tsp.context(“all”)

tsp.inittsproc(“my\_tsproc\_file.ctl”, “my\_tsproc\_recfile.txt”)

tsp.runtsproc()

# other executables could be launched before or after TSPROC

# is invoked

Figure 46 shows a Python script that still relies on the traditional TSPROC control file for direction, but compares the current block name to the name of a routine implemented in Python. If the appropriate block name is found, it is executed within the Python script, a new series or table is created and added to the list of series or tables in TSPROC memory, and processing of the control file resumes.

1. Example Python script that scans blocks in a TSPROC control file, processing the block that it has a routine for.

import pytsproc.main\_loop as tsp

tsp.context(“all”)

tsp.inittsproc(“my\_tsproc\_file.ctl”, “my\_tsproc\_recfile.txt”)

blockname = tsp.getnextblockname()

while blockname.strip().upper() != “EOF”:

if(blockname.strip().upper() == “MY\_NEW\_TSPROC\_FUNCTION”:

(keywords, arguments) = tsp.getblock()

# make a call to a new, custom function

# custom function must:

# - process keywords and arguments

# - call tsp.getseries or tsp.gettable in order to obtain

# locally modifiable copies of the series or table

# - process the series or table accordingly

# - call tsp.makeseries or tsp.maketable to add the newly

# created series or table to the series or tables that

# the TSPROC Fortran module holds in memory

myNewTSPROCFunction(keywords, arguments)

else:

# this is a block that presumably can be dealt with by the

# Fortran modules…send control back

tsp.processblock()

Figure 47 shows an example of a Python script that does not need a TSPROC control file at all to direct its processing. Instead, the script creates “in-memory” TSPROC blocks on-the-fly and causes them to be processed by TSPROC as though they originated within a control file.

1. Example of Python script that creates TSPLOT blocks in-memory.

import pytsproc.main\_loop as tsp

tsp.context(“all”)

# assemble a TSPROC block to read a series from an SSF file

tsp.newblock(“GET\_MUL\_SERIES\_SSF”)

tsp.addtoblock(“CONTEXT all”)

tsp.addtoblock(“FILE 05406500.ssf”)

tsp.addtoblock(“NEW\_SERIES q6500i”)

# now that the block has been created, make TSPROC act on it

tsp.processblock()

# assemble a TSPROC block to process a time series using the

# USGS\_HYSEP routines

tsp.newblock(“USGS\_HYSEP”)

tsp.addtoblock(“CONTEXT all”)

tsp.addtoblock(“SERIES q6500i”)

tsp.addtoblock(“NEW\_SERIES q6500\_hysp”)

tsp.addtoblock(“HYSEP\_TYPE sliding\_interval”)

# now that the block has been created, make TSPROC act on it

# TSPROC routine

tsp.processblock()

From within a Python script, a block may be built up by calling the “newblock” method first, specifying the name of the block to be created, followed by repeated calls to the “addtoblock” method. The block keywords and arguments should be given exactly as specified in the main body of this document. Finally, the “processblock” method should be called; this hands control back to the Fortran TSPROC library, with the appropriate TSPROC routines called as usual.

At first it may not appear terribly useful to be able to piece together the bits of a TSPROC block inside a Python script before having TSPROC act on it. What makes this potentially extremely useful is that it is relatively easy to have Python repeat the same actions for any number of files or time series with almost no additional lines of code. Figure 48 gives an example of a Python script that makes a list of all site sample files (SSF) within the current directory, reads and creates a TSPROC time series for each, and then performs a hydrograph separation for each.

1. Example of Python script that processes all SSF files in the current directory.

import os

import pytsproc.main\_loop as tsp

# define a function which creates a GET\_MUL\_SERIES\_SSF block

def getmulseriesssf(filename, series):

tsp.newblock(“GET\_MUL\_SERIES\_SSF”)

tsp.addtoblock(“CONTEXT all”)

tsp.addtoblock(“FILE “ + filename)

tsp.addtoblock(“NEW\_SERIES ”+ series)

tsp.processblock()

# define a function which creates a USGS\_HYSEP block

def usgshysep(series):

tsp.newblock(“USGS\_HYSEP”)

tsp.addtoblock(“CONTEXT all”)

tsp.addtoblock(“SERIES “ + series)

tsp.addtoblock(“NEW\_SERIES “ + series + “\_sep”)

tsp.addtoblock(“HYSEP\_TYPE sliding\_interval”)

tsp.processblock()

# get a list of all files in current directory

filenames = os.listdir(os.curdir)

# for all \*.ssf files in the current directory, read in the

# file and process the series using the USGS Hysep methods

for filename in filenames:

if ‘ssf’ in filename:

# split the filename; name the series after the file prefix

series = filename.split(“.”)[0]

# read the SSF file, create the series

getmulseriesssf(filename, series)

# run hydrograph separation on the series

usgshysep(series)

The concept employed in Figure 48 may be used to process an almost unlimited number of files without increasing the size of the script; a dozen or mode lines of TSPROC keyword and argument pairs might be needed for each additional SSF file processed by means of the traditional control file approach.

1. Watermark Numerical Computing, Brisbane, Australia [↑](#footnote-ref-1)
2. Layne Hydro, Bloomington, Indiana, United States [↑](#footnote-ref-2)