###### TS6PROC

###### A Time Series Processor

###### for

###### MODFLOW 6

**John Doherty**

Watermark Numerical Computing

July, 2020

Acknowledgements

I would like to thank BHP and Rio Tinto. These companies fund the GMDSI project which has contributed significantly to the development of this software. GMDSI stand for “Groundwater Modelling Decision Support Initiative”. This 3.5 year project, conducted under the auspice of National Centre for Groundwater Research and Training (NCGRT) at Flinders University, Australia, commenced in late 2019. Its aim is to create greater awareness of issues associated with decision-support groundwater modelling, and to assist in addressing these issues through dialogue, education, worked examples, research and strategic software development.

John Doherty

Table of Contents

[1. Introduction 1](#_Toc43972644)

[1.1 General 1](#_Toc43972645)

[1.2 Running TS6PROC 1](#_Toc43972646)

[2. TS6PROC Control File 2](#_Toc43972647)

[2.1 TS6PROC Processing 2](#_Toc43972648)

[2.2 An Example 2](#_Toc43972649)

[2.3 The Files Block 3](#_Toc43972650)

[2.4 The Parameters Block 3](#_Toc43972651)

[2.5 The Processing Block 4](#_Toc43972652)

[3. Equations 6](#_Toc43972653)

[3.1 General 6](#_Toc43972654)

[3.2 Operators and Mathematical Functions 6](#_Toc43972655)

[Arithmetic operators 6](#_Toc43972656)

[Logical operators 6](#_Toc43972657)

[Mathematical functions 7](#_Toc43972658)

[3.3 Equation Assignments 8](#_Toc43972659)

[3.4 Some Examples 9](#_Toc43972660)

[Equations involving scalars 9](#_Toc43972661)

[Equations involving time series 9](#_Toc43972662)

[4. TS6PROC Functions **Error! Bookmark not defined.**](#_Toc43972663)

[4.1 General **Error! Bookmark not defined.**](#_Toc43972664)

[4.2 Functions **Error! Bookmark not defined.**](#_Toc43972665)

[term\_average() **Error! Bookmark not defined.**](#_Toc43972666)

[time\_average() **Error! Bookmark not defined.**](#_Toc43972667)

[expand\_about\_value() **Error! Bookmark not defined.**](#_Toc43972668)

[copy\_terms() **Error! Bookmark not defined.**](#_Toc43972669)

# 1. Introduction

## 1.1 General

TS6PROC stands for “Time Series 6 Processor”.

TS6PROC reads a MODFLOW 6 time series file. It manipulates time series that are contained in this file in ways that are specified through a TS6PROC input control file. It then writes a new MODFLOW 6 time series file that contains the altered time series. This altered file can be used by MODFLOW 6 in place of the original one.

TS6PROC offers a number of options for construction and manipulation of time series. A time series can be built and/or modified through application of equations of arbitrary complexity that apply to all terms of existing series. Alternatively, these equations can be used to build a new time series with the same time base as existing time series. These equations can feature parameters that are manipulated by members of the PEST and/or PEST++ suites when undertaking model calibration and/or calibration-constrained uncertainty analysis. TS6PROC also provides a number of special functions for manipulation of time series. The number of these functions is expected to increase over time.

## 1.2 Running TS6PROC

TS6PROC can be run in either of two ways. The first way it to type its name at the command prompt:

ts6proc

Immediately upon commencement of execution TS6PROC prompts for the name of its control file:

Enter name of TS6PROC input control file:

Provide the name of this file as requested. Surround its name by quotes if it contains a space.

Alternatively, the name of the TS6PROC control file can be provided on the TS6PROC command line. Suppose that the name of this file is *file.ctl*. Then TS6PROC can be run using the command:

ts6proc file.ctl

The contents of a TS6PROC control file are described in the next chapter.

As TS6PROC runs, it informs the user of its processing activities through its screen output. Just before it terminates execution, it writes the new MODFLOW 6 time series file containing replacement time series. Alternatively if, at any stage of its execution, it finds an error in its input dataset, it reports this error to the screen, and then immediately ceases execution.

# 2. TS6PROC Control File

## 2.1 TS6PROC Processing

TS6PROC reads a MODFLOW 6 time series file (henceforth referred to as a TS6 file). It writes a new TS6 file in which the terms of one or a number of time series represented in the first file are altered from their original values. The new time series file contains the same time series as does the original file; the number of terms in these series, and the names of these series, are unaltered. Hence a MODFLOW 6 model encounters no problems when reading this altered file in place of the original time series file.

The name of the TS6 file that it must read, and the name of the TS6 file that it must write, are listed in a TS6PROC control file. So too are specifications of the processing steps that time series in the first file must undergo before being recorded in the new file.

## 2.2 An Example

Figure 2.1 shows an example of a TS6PROC input control file. This example forms the basis for the following discussion.

|  |
| --- |
| # Here is a comment  BEGIN FILES  FILEIN river\_stages\_orig.ts  FILEOUT river\_stages1.ts  END FILES  BEGIN PARAMETERS  genmul1 = 1.1  genmul2 = 1.1  expand\_factor1 = 1.0  expand\_factor2 = 1.0  END PARAMETERS  BEGIN PROCESSING  river\_stage\_1 = river\_stage\_1 \* genmul1  river\_stage\_2 = river\_stage\_2 \* genmul2  average1 = time\_average(river\_stage\_1,'log')  average2 = time\_average(river\_stage\_1,'log')  river\_stage\_1 = expand\_about\_value(river\_stage\_1,average1,expand\_factor1,0.0,100.0)  river\_stage\_1 = expand\_about\_value(river\_stage\_2,average2,expand\_factor2,0.0,100.0)  END PROCESSING |

**Figure 2.1. Example of a TS6PROC input control file.**

As is apparent from figure 2.1, a TS6PROC control file is divided into blocks. These blocks are named as follows:

* files
* parameters
* processing

These three blocks must be provided in the order shown above. In accordance with protocols adopted by MODFLOW 6, the start of a block is indicated by the word “BEGIN” followed by the name of the block. The “END” keyword (with or without the name of the block) denotes its end. Note that TS6PROC will accept “START” instead of “BEGIN”.

All text within a TS6PROC control file is case insensitive. Blank lines can be inserted anywhere. Any text that follows a “#” character is interpreted as a comment. The “#” character can be placed at the start of a line, or at any location within a line.

Specifications for blocks within a TS6PROC control file are now discussed in detail.

## 2.3 The Files Block

Two pieces of information are provided through the “files” block. These are

1. The name of the MODFLOW 6 time series file (i.e. TS6 file) which TS6PROC must read, and
2. The name of the MODFLOW 6 time series file which TS6PROC must write.

The first of the above filenames follows the FILEIN keyword, while the second of the above filenames follows the FILEOUT keyword. If the name of a file contains a space, it should be enclosed in quotes. The above keywords can be provided in either order.

TS6PROC reads all time series contained in the input TS6 file. The names of these time series can be used in equations and functions that are provided in the “processing” block of a TS6PROC control file. Certain rules apply to these names. In particular:

* The name of a time series must be 50 characters or less in length.
* It must not contain a space.
* The name must not collide with other names that are used by TS6PROC that have special meanings. These names include “time”, the names of TS6PROC functions, and the names of standard mathematical functions that can appear in TSPROC equations.
* The name of a time series cannot include special symbols such as “+” and “/” that are reserved for mathematical operations.

## 2.4 The Parameters Block

Each line in the “parameters” block of a TSPROC control file must contain two entries. The first is the name of a parameter, while the second is the value of that parameter. Optionally, these two entries can be separated by an “=” symbol. A “parameter” is a scalar variable that can feature in an equation or function listed in the TS6PROC “processing” block. The values assigned to parameters can be adjusted by PEST or PEST++. This will require the construction of a template file based on a TS6PROC control file. A PEST template file corresponding to the TS6PROC control file illustrated in figure 2.1 is depicted in figure 2.2.

|  |
| --- |
| ptf $  # Here is a comment  BEGIN FILES  FILEIN river\_stages\_orig.ts  FILEOUT river\_stages1.ts  END FILES  BEGIN PARAMETERS  genmul1 = $ genmul1 $  genmul2 = $ genmul2 $  expand\_factor1 = $efac1 $  expand\_factor2 = $efac2 $  END PARAMETERS  BEGIN PROCESSING  river\_stage\_1 = river\_stage\_1 \* genmul1  river\_stage\_2 = river\_stage\_2 \* genmul2  average1 = time\_average(river\_stage\_1,'log')  average2 = time\_average(river\_stage\_1,'log')  river\_stage\_1 = expand\_about\_value(river\_stage\_1,average1,expand\_factor1,0.0,100.0)  river\_stage\_1 = expand\_about\_value(river\_stage\_2,average2,expand\_factor2,0.0,100.0)  END PROCESSING |

**Figure 2.2. A template file based on the TS6PROC input control file shown in figure 2.1.**

The following protocols apply to TS6PROC parameter names.

* The name of a TS6PROC parameter can be considerably longer than that of a PEST parameter. It can be up to 50 characters in length.
* A parameter name must not contain a space.
* Certain parameter names are disallowed. “time” is disallowed, as are names that can be confused with TS6PROC functions, or standard mathematical functions that can appear in TS6PROC equations.
* Parameter names cannot include operators such as “+” and “/” that appear in equations.

## 2.5 The Processing Block

The “processing” block of a TSPROC control file lists a sequence of processing steps. A processing step is encapsulated in an equation or by a TS6PROC function (but not both). The equation or function must be placed on the right of an “=” sign. The name of the variable to which the outcomes of that processing are assigned is placed on the left of the “=” sign. This can be an “implicit parameter” or a user-supplied or implicit time series.

As the name suggests, an implicit parameter or time series is brought into existence by an equation or function which follows an “=” symbol. This occurs if the name on the left side of the equation has not been previously defined. TS6PROC works out for itself whether the entity which it creates is a parameter or a time series. Once this entity has been defined and assigned a value through the processing step (i.e. through the equation of function which lies to the right of the “=” symbol), it can be used on the right side of future processing steps as the argument of a function or as a term of an equation.

It is important to note, however, that TS6PROC will not allow the values of parameters which are defined in its “parameters” section to be over-written by functions or equations. This safeguards their values, as assigned by a user or by PEST. Hence these parameters (referred to herein as “explicit parameters”) must not appear on the left side of any operation defined in the “processing” block of a TS6PROC control file.

The same does not apply, however, to time series that are read from a TS6 file. These can be over-written by processing operations defined in a TS6PROC control file. In fact, if the name of such a time series is not featured on the left of at least one equation or function, then the running of TS6PROC has no purpose. Recall that its job is to modify time series appearing in a TS6 file, and to re-write that file using these modified time series.

Equations and functions are described in detail in the following sections of this document. At this point it is salient to note that if one or more time series appear in an equation, then the outcome of that equation is a time series. In this case the mathematical operations defined by the equation are successively applied to every term of all time series that appears on its right hand side. However if the value associated with a term of any of the time series is the MODFLOW 6 “no data” value (i.e. 3.0E30), then this value is transferred directly to the corresponding term of the time series that appears on the left side of the equation.

If an equation or function provided in the “processing” block of a TS6PROC control file is too long to represent conveniently on a single line, then it can be spread over several lines. If the last character appearing on any line provided in the “processing” block of a TS6PROC control file is “&”, then TS6PROC will append the following line to this line before processing it.

# 3. Equations

## 3.1 General

Much of the present section of this document is extracted from the manual for PLPROC. The equation functionality provided by TS6PROC is very similar to that provided by PLPROC.

As stated in the previous section of this document, an equation can be used to assign a value to a single variable (i.e. to a parameter), or to terms of a time series. In the latter case, TS6PROC performs the same operation successively on each term of the time series that are featured in the equation. Because TS6PROC reads time series from a user-supplied TS6 file, all of the time series in its memory have the same time base. Times comprising this base are listed in the first column of the TIMESERIES block of the TS6 input file which TS6PROC reads.

It is important to note that while equations supplied to TS6PROC can contain both operators and mathematical functions (see below), they cannot contain special TS6PROC functions (see the next section of this document). If a special TS6PROC function appears in an equation, it must be the only term in this equation; that is, it must appear on its own immediately following an “=” sign on a single line of the TS6PROC “processing” block.

## 3.2 Operators and Mathematical Functions

### Arithmetic operators

Table 3.1 shows a list of arithmetic operators that can appear in a TS6PROC equation. Note that the usual rules of operator precedence apply. If you are in doubt about precedence, use brackets to enforce your own order; inner bracketed expressions are always evaluated first.

|  |  |
| --- | --- |
| **Operator symbol** | **Operation** |
| \*\* *or* ^ | Power. *a*\*\**b* or *a*^*b* is interpreted as “*a* raised to the power *b*”. |
| / | Division. *a*/*b* is interpreted as “*a* divided by *b*”. |
| \* | Multiplication. *a*\**b* is interpreted as “*a* multiplied by *b*”. |
| - | Subtraction. This can be a unary or binary operator. *a*-*b* is interpreted as “*a* minus *b*”; -*a* is interpreted as “negative *a*”. |
| + | Addition. This can be a unary or binary operator. *a*+*b* is interpreted as “*a* plus *b*”; +*a* is interpreted as “positive *a*”. |
| ( ) | Brackets. Terms within brackets are evaluated first. For example  5 + 4 \* 3 is evaluated as 17. However (5 + 4) \* 3 is evaluated as 27. |

Table 3.1 Arithmetic operators supported by TS6PROC equations.

### Logical operators

Logical operations produce an outcome that is either *TRUE* or *FALSE*. Logical operators supported by TS6PROC are listed in table 3.2.

|  |  |
| --- | --- |
| **Operator symbol** | **Operation** |
| .lt. *or* < | Less than. *a*.lt*.b* or *a*<*b* is *TRUE* if *a* is less than *b*. |
| .le. *or* <= | Less than or equal to. *a.*le.*b* or *a*<=*b* is *TRUE* if *a* is less than or equal to *b*. |
| .eq. *or* == | Equal to. *a.*eq*.b* or *a*==*b* is *TRUE* if *a* equals *b*. |
| .gt. *or* > | Greater than. *a.*gt.*b* or *a*>*b* is *TRUE* if *a* is greater than *b*. |
| .ge. *or* >= | Greater than or equal to. *a.*ge.*b* or *a*>=*b* is *TRUE* if *a* is greater than or equal to *b*. |
| .ne. *or* != | Not equal to. *a*.ne.*b* or *a*!=*b* is *TRUE* if *a* does not equal *b*. |
| .and. o*r* && | And. *a.*and.*b* or *a*&&*b* is *TRUE* if both *a* and *b* are true; for example (1.lt.10).and.(6.lt.7) is *TRUE*. |
| .or. *or* || | Or. *a*.or***.****b* or *a*||*b* is *TRUE* if *a* is *TRUE* or *b* is *TRUE* or both are *TRUE*; for example (1.lt.10).or.(1.gt.0) is *TRUE*. |
| ! | Not. !(*a*.lt*.b*) is *FALSE* if *a* is less than *b*. |

Table 3.2 Logical operators supported by TS6PROC.

As has already been discussed, the outcomes of TS6PROC equations are assigned to implicitly defined parameters or to elements of an explicitly or implicitly defined time series. Both of these data types host values that are real numbers. If a TS6PROC equation with a logical outcome makes a value assignment to one of these entities, then *TRUE* is converted to 1.0 and *FALSE* is converted to 0.0.

### Mathematical functions

TS6PROC equations support the following mathematical functions. The outcomes of all of these equations are real numbers.

|  |  |
| --- | --- |
| **Function** | **Definition** |
| abs( ) | Absolute value. Argument can be any real number. |
| cos( ) | Cosine. Argument can be any real number supplied in radians. |
| acos( ) | Inverse cosine. Absolute value of argument must be less than or equal to one. Value is returned in radians. |
| sin( ) | Sine. Argument can be any real number supplied in radians. |
| asin( ) | Inverse sine. Absolute value of argument must be less than or equal to one. Value is returned in radians. |
| tan( ) | Tan. Argument can be any real number supplied in radians. |
| atan( ) | Inverse tan. Argument can be any real number. Value is returned in radians. |
| cosh( ) | Hyperbolic cosine. Argument can be any real number. |
| sinh( ) | Hyperbolic sine. Argument can be any real number. |
| tanh( ) | Hyperbolic tan. Argument can be any real number. |
| exp( ) | Exponential. Argument can be any real number. |
| log( ) | Log to base *e*. Argument must be a positive real number. |
| log10( ) | Log to base 10. Argument must be a positive real number. |
| sqrt( ) | Square root. Argument must be non-negative. |
| min( , , ) | Minimum of a series of numbers. Arguments can be any real numbers. |
| max( , , ) | Maximum of a series of numbers. Arguments can be any real numbers. |
| mod( , ) | Remainder. mod(*a*,*b*) is the remainder after *a* is divided by *b*. |

Table 3.3 Mathematical functions supported by TS6PROC equations.

If an equation requests an impossible operation (for example the square root of a negative number) TS6PROC ceases its attempts to evaluate the equation; instead, it terminates execution with an appropriate error message.

## 3.3 Equation Assignments

If a processing step is recognized as an equation (rather than as a call to a TS6PROC special function), TS6PROC looks for the names of variables that are cited in the equation as it reads it. If any of these variables is a time series (or the variable *time*), then the outcome of the equation must also be a time series. If, instead, the entity on the left of the equation is an explicitly or implicitly defined parameter, then an error conditions arises. On the other hand, if the entity on the left of the equation is a time series, then terms of this time series are populated using the equation. Alternatively, if the entity on the left of the “=” sign has not been previously defined, then TS6PROC defines it as a time series; its terms are then populated using the equation.

If all entities cited on the right side of an equation are parameters, but the entity on the left of the equation is a previously-defined, explicit or implicit time series, then all terms of the time series are populated using the equation; in this case, they are all assigned the same value. Alternatively, if the entity on the left of an equation is an explicit parameter defined in the “parameters” section of the TS6PROC control file, then TS6PROC ceases execution with an error message, as the values of parameters defined in this way cannot be over-written. However if the entity on the left of the “=” sign is an implicitly defined parameter, its value is re-assigned through evaluation of the equation. Alternatively, if the entity on the left of the equation has not been previously defined, it is created as an implicit parameter, and assigned a value through evaluation of the equation.

## 3.4 Some Examples

### Equations involving parameters

Figure 3.1 shows a sequence of equations which involve only parameters. Let us suppose that *p1* and *p2* are parameters that are defined explicitly in the “parameters” section of a TS6PROC control file, or have been assigned values implicitly through being cited on the left side of previous equations. Parameters *p3* and *p4* may be defined through the above equations; alternatively, they may have been defined implicitly through previous equations.

|  |
| --- |
| .  p3=4.49234  p3=p3\*(p1^2+p2^2)  p4=p3  . |

Figure 3.1 Portion of a TS6PROC processing block showing equations involving parameters.

Note how a parameter (*p3* in the above case) can appear on both the left and right sides of an equation, as in any programming language.

If the first of the equations depicted in figure 3.1 constitutes the first mention of *p3*, then a parameter named *p3* is brought into existence by the equation. The same applies to *p4* in the third of the equations depicted in figure 3.1. On the other hand, if implicit parameters *p3* and *p4* already exist, then their existing values are replaced by equation-calculated values.

### Equations involving time series

Now consider the TS6PROC equations depicted in figure 3.2. Let us suppose that *s1* is a time series which has either been read from a TS6 file, or has been previously defined implicitly using equations. Meanwhile, *p1* and *p2* are parameters.

|  |
| --- |
| .  s1=6322.11  new\_s = p1/p2\*4.321 + s1  s1=new\_s+p1  s1=s1 + sin(time\*2\*pi/365.25)  . |

Figure 3.2 Portion of the “processing” section of a TS6PROC control file showing equations involving parameters and time series.

If *s1* had not already been defined as a time series, it would be defined implicitly as a parameter through the first of the above equations, as this equation cites no time series, nor the entity *time*, on its right hand side. However with *s1* previously defined as a time series, the first equation endows all of its terms with a value of 6322.11.

The right side of the second of the above equations features parameters and a time series. Hence the left side of the equation must be a time series. If *new\_s* does not already exist as a time series, it is defined as such through this equation, at the same time as its terms are populated with numbers through repeated evaluation of the equation.

Terms of the *s1* time series are assigned new values through repeated evaluation of the third of the above equations. They are assigned new values again through the fourth equation. This equation embodies addition of a sine wave with a period of 365.25.

As was stated in the previous section of this document, if any term of a time series that appears on the right side of an equation has a value of “no data”, then the corresponding term on the left of the equation is assigned this same value, regardless of the values of terms belonging to other time series that are cited on the right of the equation.

# 4. TS6PROC Functions

## 4.1 General

TS6PROC supports a number of special functions that process time series in ways that are difficult to process through more general equations, or that assign a value to a parameter based on the contents of one or more time series. In the latter case, the (necessarily implicitly-defined) parameter can be used in TS6PROC functions and equations which appear in ensuing processing steps.

As has been previously discussed, a TS6PROC special function is invoked through its appearance immediately to the right of an “=” sign on a line of the “processing” section of a TS6PROC input control file. The name of the function must be followed by an opening bracket. Arguments of the function, separated by commas, must follow the opening bracket. A closing bracket must follow these arguments. Nothing must follow the closing bracket. Hence a TS6PROC function cannot be part of a more general equation.

As is documented below, arguments to TS6PROC functions can be either time series or parameters. Numbers can be used in place of parameters if desired. Function arguments must be provided to a TS6PROC function in the order shown in their documentation.

The number of functions provided by TS6PROC is likely to increase over time. Naturally, this manual describes only those that are available at the time of writing.

Each of the functions that are presently supported by TS6PROC is now described in detail.

## 4.2 Functions

### term\_average()

#### Description

Function *term\_average()* computes the average of terms of a time series. Optionally, the averaging process can take place in the domain of natural logarithms; in this case the outcome of the averaging process is exponentiated before assignment to the entity on the left of the equation. The latter must be a parameter. It can be defined implicitly through use of the function; alternatively, it must have been defined implicitly through use of a previous function or equation.

The averaging process ignores “no data” values. The average is calculated as the sum of the terms of the time series (excluding “no data” terms), divided by the number of elements over which summation takes place.

#### Example

par6 = term\_average(series,log)

#### Arguments

|  |  |  |
| --- | --- | --- |
| **Argument number** | **Type of entity** | **Specifications** |
| 1 | time series | Terms must be greater than zero if the second argument is “log”. |
| 2 | character string | Must be “none” or “log”. |

#### Notes

1. The text string “none” or “log” comprising the second argument of function *term\_average()* can optionally be surrounded by quotes in a call to this function.
2. If the second argument of function *term\_average()* is “log”, then terms of the time series are logged before averaging. If any term is zero or negative, TS6PROC ceases execution with an error message.

### time\_average()

#### Description

Function *time\_average()* computes the average of terms of a time series with time taken into account. It uses the trapezoidal rule to integrate time series values over time. It divides by the length of the time series (calculated as the time interval between the first and last time) to obtain the average. The result is assigned to a parameter. Time series elements with “no data” values are ignored, as are the times with which they are associated.

Optionally, the natural log can be taken of time series values before integration and averaging. If the average is evaluated in this way, it is exponentiated before being assigned to a parameter.

#### Example

par6 = time\_average(series,log)

#### Arguments

|  |  |  |
| --- | --- | --- |
| **Argument number** | **Type of entity** | **Specifications** |
| 1 | time series | Terms must be greater than zero if the second argument is “log”. |
| 2 | character string | Must be “none” or “log”. |

#### Notes

1. The text string “none” or “log” comprising the second argument of function *time\_average()* can optionally be surrounded by quotes in a call to this function.
2. If the second argument is “log”, then terms in the time series are logged before integration and averaging. If any term is zero or negative, TS6PROC ceases execution with an error message.

### time\_average\_over\_interval()

#### Description

Function *time\_average\_over\_interval()* performs a similar task to that performed by function *time\_average()*. However averaging takes place over a user-specified time interval. This interval is defined by two times. Note, however, that *time\_average\_over\_interval()* does not interpolate or extrapolate to these user-specified times. It commences the averaging period at the first time cited in the time series file that is equal to or post-dates the first user-supplied time. It terminates the averaging time interval at the last time in the time series that is equal to, or precedes, the second user-specified time. The MODFLOW 6 “no-data” value (i.e. 3.0E30) is taken into account when defining the averaging interval. Times associated with “no-data” values are treated as absent from the time series; the no-data values are, or course, ignored.

#### Example

average = time\_average\_over\_interval(rivstage,none,99.0,1001.0)

#### Arguments

|  |  |  |
| --- | --- | --- |
| **Argument number** | **Type of entity** | **Specifications** |
| 1 | time series | Terms within the averaging time interval must be greater than zero if the second argument is “log”. |
| 2 | character string | Must be “none” or “log”. |
| 3 | parameter or number | The start of the averaging time interval is the first time cited in the time series that equals or follows this time. |
| 4 | parameter or number | The end of the averaging time interval is the last time cited in the time series that equals or precedes this time. |

#### Notes

1. The text string “none” or “log” comprising the second argument of function *time\_average\_over\_interval()* can optionally be surrounded by quotes.
2. If the second argument is “log”, then terms in the time series are logged before integration and averaging. If any terms within the user-nominated time interval are zero or negative, TS6PROC ceases execution with an error message.

### expand\_about\_value()

#### Description

Function *expand\_about value()* calculates the terms of one time series from those of another. It does this in a number of steps.

Suppose that the user supplies the value *a* for a number about which series expansion takes place. On most occasions on which this function is called, this number will have been computed using function *time\_average()* or *term\_average()*. For each element *ei* of the time series, function *expand\_about\_value()* calculates a “residual” *ri* as:

*ri* = *ei* – *a*

This residual is then multiplied by a user-supplied factor *f*. It is then added back to *a* to evaluate the new value for the time series term. If the new value of any term of the expanded time series is less than a user-supplied lower bound *b*, it is clipped at *b*. If it is greater than a user-supplied upper bound value *u*, it is clipped at *u*.

#### Example

newseries = expand\_about\_value(series,a,f,lbound,rbound)

#### Arguments

|  |  |  |
| --- | --- | --- |
| **Argument number** | **Type of entity** | **Specifications** |
| 1 | time series | An explicitly or implicitly-defined time series. |
| 2 | parameter or number | The value about which series expansion takes place (*a* in the above description). |
| 3 | parameter or number | The expansion factor (*f* in the above description). |
| 4 | parameter or number | Lower bound of the expanded time series (*b* in the above description). |
| 5 | parameter or number | Upper bound of the expanded time series (*u* in the above description). |

### copy\_terms()

#### Description

The outcome of function *copy\_terms()* is a time series. As usual, this can be an existing time series, or a time series that is created through evaluation of the function. This output time series is formed by copying terms from one time series over one time interval, while copying terms from another time series over the remainder of the time interval spanned by all time series. Where the output time series is one of the time series from which copying is performed, this amounts to partial series replacement of terms of that time series by terms of the other time series.

#### Example

ts1 = copy\_terms(ts1,ts2,100.0,300.0)

#### Arguments

|  |  |  |
| --- | --- | --- |
| **Argument number** | **Type of entity** | **Specifications** |
| 1 | time series | Terms are copied from this time series outside the user-defined time period. |
| 2 | time series | Terms are copied from this time series inside the user-defined time period. |
| 3 | parameter or number | A time equal to or greater than the beginning of the time series. |
| 4 | parameter or number | A time equal to or less than the end of the time series. |

#### Notes

1. The 3rd and 4th arguments of function *copy\_terms()* define a time period. Within this time period, terms from the second time series are copied to the output time series. Outside of that time period, terms from the first time series are copied to the output time series.
2. Although the function allows it, it would be unusual for the 3rd and 4th argument of function *copy\_terms()* to be parameters rather than numbers. It would be difficult to estimate these parameters during an inversion process, as alterations to their values would cause discontinuous changes to the output time series.

### assign\_terms()

#### Description

The outcome of function *assign\_terms()* is a time series. As usual, this can be an existing time series, or a time series that is created through evaluation of the function. This output time series is formed by assigning a user-specified value to terms of the series over one time interval, and by copying terms from another time series within the remaining time interval. Where the output time series is the time series from which copying is performed, this amounts to partial series replacement of terms of that time series by the user-specified value.

#### Example

ts1 = assign\_terms(ts1,value,100.0,300.0)

#### Arguments

|  |  |  |
| --- | --- | --- |
| **Argument number** | **Type of entity** | **Specifications** |
| 1 | time series | Terms are copied from this time series outside the user-defined time period. |
| 2 | parameter or number | Terms are assigned this value inside the user-defined time period. |
| 3 | parameter or number | A time equal to or greater than the beginning of the time series. |
| 4 | parameter or number | A time equal to or less than the end of the time series. |

#### Notes

1. The 3rd and 4th arguments of function *assign\_terms()* define a time period. Within this time period, terms of the output time series are assigned a value equal to the 2nd argument. Outside of that time period, terms from the time series supplied as the first argument are copied to the output time series.
2. Although the function allows it, it would be unusual for the 3rd and 4th argument of function *assign\_terms()* to be parameters rather than numbers. It would be difficult to estimate these parameters during an inversion process, as alterations to their values would cause discontinuous changes to the output time series.