triwaco

groundwater modelling software

C Allocators and Geo-processing

ROYAL HASKONING

Appendix C: Allocators and geo-processing

C.1 Introduction

Triwaco includes a range of powerful geo-processors for 1D to 4D interpolation. The processors are called allocators since they are used to assign (allocate) parameter GIS maps/values to the individual nodes or cells of the grid. Most allocators can be used for different types of parameters. For source, river and boundary parameters specific allocators are available. Other allocators are used for distributed parameters only (assigning a parameter value to each node of the grid).

There are two types of allocators available: **Internal allocators** and **External allocators**.

The Internal allocators (None, Const, Expression, InvDist, TinInterpol, ParBou, SrcParAdo, ParRiv and ParAdo) are included in the **TriShell** itself. These allocators cannot be added or deleted, nor run as a stand alone program.

The External allocators (xConst, xParAdo, xParBou, XparRiv, xSrcParAdo, xInvDist, xTin, Kriging, Arpadi, LabP2Ado, Ado2RBS, TSA, Flo2Ado, AllocExp, DosBatch) used from the TriShell but alos run as a stand alone program. The modular structure of Triwaco also applies for the allocators. An external allocator can be added by choosing 'Add' and specifying the appropriate name and description in the dialog box (see chapter 2).

A special allocator is the Expression-allocator. The expression allocator evaluates mathematical expressions between given parameters. It can alos be used to convert river/node/source parameters to river/node/source parameters.

The following allocators (internal and external) are included in the standard **Triwaco** package:

Internal allocators		
Allocator name	**Allocation method**	**when to use**
None	No allocation; the parameter has to be given in ***Adore*** format.	Use for **any** parameter for which an ***Adore***-set is available.
Const	Assignment of a single value over the whole domain; input is defined by a constant, given in the ***'parameter info screen'***.	Use for **any** parameter having one single value only.
ParAdo	Assignment of parameter values read from and defined by a single par file (ID and a value).	Use for **any** parameter having only a parameter file. Note that the number of values in the ***Adore***-set depends on the maximum ID encountered.
ParBou	Linear interpolation of parameter values to boundary nodes; input is defined by 'linked points' with coordinates and a value.	Use for **Boundary** parameters only.
ParRiv	Linear interpolation of parameter values to river nodes; input is defined by linked points with river-ID, coordinates and a value.	Use for **River** parameters only.
SrcParAdo	Assignment of parameter values for source nodes defined by source-ID and a value. The number of sources and their IDs are read from the grid	Use for **Source** parameters only. Note only assigns values for IDs encountered in the parameter file; other sources are assigned the default value.
Expression	Evaluates mathematical expressions between given parameters. The expression has to be defined in the ***'parameter info screen'***.	Use for **any** parameter. Note that the sets the expression points to should be regular ***Adore***-sets.
InvDist	Inverse Distance allocator for distributed parameters; input defined by point-values.	Use for any **distributed** parameter.
TinInterpol	Interpolation by Triangulation for distributed parameters; input defined by point-values, which cover the total model area.	Use for any **distributed** parameter.

External allocators		
Allocator name	**Allocation method**	**when to use**
Kriging	Interpolates values defined by polygons, lines and points.	Use for a distributed parameter.
Arpadi	Assignment of a constant value for areas defined by polygons, or interpolation using Thiessen-polygons for point-values.	Use for a **any** parameter.
ArpLin	Lineair interpolation through three points	Use for a **any** parameter
ArpMuQ	Multi-Quadratic interpolation	Use for a **any** parameter
ArpTie	Thiessen polygons	Use for a **any** parameter
LabPar2Ado	Converts a *.lab (or *.arc) with a *.par (or *.aci) file to an adorefile	
Ado2RBS	Converts a node-adorefile (specified as *.ung in the parameterinfo box) to a River, Source or Boundary parameter.	Redundant since version 3.0. Incorperated in the expression evaluator.
ConvAdo	Converts a river/node/source-adorefile (specified as *.ung in the parameterinfo box) to a River, Source or Boundary parameter.	Redundant since version 3.0. Incorperated in the expression evaluator.
TSA	Fast alternative for timedependant allocation of polygons to nodes	
Flo2Ado	Copies a random set from the timedependant *.flo file	
AllocExp	Uses simple expressions for use with TrCalCon	
DosBatch	Runs an user-made allokator as a batch file	Run batch files in windows mode
xConst	Assignment of a single value over the whole domain; input is defined by a constant, given in the ***'parameter info screen'***.	Use for **any** parameter having one single value only.
xParAdo	Assignment of parameter values read from and defined by a single par file (ID and a value).	Use for **any** parameter having only a parameter file. Note that the number of values in the ***Adore***-set depends on the maximum ID encountered.
xParBou	Linear interpolation of parameter values to boundary nodes; input is defined by 'linked points' with coordinates and a value.	Use for **Boundary** parameters only.
xParRiv	Linear interpolation of parameter values to river nodes; input is defined by linked points with river-ID, coordinates and a value.	Use for **River** parameters only.
xSrcParAdo	Assignment of parameter values for source nodes defined by source-ID and a value. The number of sources and their IDs are read from the grid	Use for **Source** parameters only. Note only assigns values for IDs encountered in the parameter file; other sources are assigned the default value.
xInvDist	Inverse Distance allocator for distributed parameters; input defined by point-values.	Use for any **distributed** parameter.
xTinInterpol	Interpolation by Triangulation for distributed parameters; input defined by point-values, which cover the total model area.	Use for any **distributed** parameter.
xAdoPar		
GrWindow	Interpolate values from a larger model	Use when a window model is created for **distributed** parameters and no paramter maps are available
SGrWindow	Assign values from a larger model from the nearest node/cell	Use when a window model is created for **distributed** parameters
Mv2RP13	Assignement of RP13 (surface level in topsystem)	Use for filtering of depressions in the file MV.ado
RivAlloc	Assignment of river and topsystem parameters	Definition of surface water system (advanced users only)

Note that the ***External allocators*** xConst, xParAdo, xParBou, xParRiv, xSrcParAdo, xInvDist, xTin are virtually the same as the ***Internal allocators*** Const, ParAdo, ParBou, ParRiv, SrcParAdo, InvDist and TinInterpol. Defining these allocators externally, however, allows the user to use some special features, whereas the ***Internal allocators*** use a default set of options.

C.2 Internal Allocators

C.2.1 None
Certain parameters may be allocated using a GIS, a user-defined procedure or will be adopted from former **Triwaco** models. These parameters do not need to be allocated. The allocator is set to **None**; the result file has to be a standard ***Ado***-file.

C.2.2 Const
Some parameters may need only one overall value for all nodes. This can be accomplished by using the **Const** (Constant) allocator. This allocator will produce a result file with only one value for all nodes. This value equals the default value given in the ***'Parameter Info'*** dialog box for the parameter considered. The allocator may be used for distributed parameters as well as for river, source or boundary parameters. (External equivalent XConst).

C.2.3 ParAdo
ParAdo is the internal allocator of **Triwaco** that assigns values to rivers, sources or grid nodes when only a single parameter file is available. This parameter file should contain the ID and the parameter value for at least a sub-set of the item considered. (External equivalent XParAdo).
Note that the number of values in the ***Adore***-set depends on the maximum ID encountered. This allocator may be used effectively if values are interpolated using a GIS and written to a single parameter file.
It also may be used to assign the value of the river activity parameter **RA*i***. In that case the file needed contains the river IDs and the river activity value (which is either **0**, if disabled, **1**, if active, or **2** if the river is 'clustered' to other rivers).

C.2.4 ParBou
ParBou is the internal allocator of **Triwaco** that assigns values to the boundary nodes. The allocator needs a map file, which contains linked points with the Boundary's ID (which is 1 by default) and values for the Boundary Parameter considered. Between the input data linear interpolation of parameter values is applied. (External equivalent XParBou).

C.2.5 ParRiv
ParRiv is the internal allocator of **Triwaco** that assigns values to the river nodes. The allocator needs a map file, which contains lines with linked points or linked points only. The linked points must have the river's ID and a value for the River Parameter considered. Between the input data linear interpolation of parameter values is applied. (External equivalent XParRiv).

C.2.6 SrcParAdo
SrcParAdo is the default internal allocator of **Triwaco** that assigns values to the source nodes. The allocator needs a parameter file with the ID of the sources and the value of the source parameter. The source ID is read from the grid-file. (External equivalent XSrcParAdo).

C.2.7 Expression
The expression allocator assigns the result of an expression to the nodes. The expression has to be defined in the ***'Parameter Info'*** dialog box. The allocator may be used for all types of parameters, provided the parameters referred to in the expression are of the same type. For parameters of different type a conversion of the type may be needed, for which the expression allocator provides a number of conversion-functions.

C.2.8 InvDist
The InvDist allocator interpolates values given as point-data. The interpolation method calculates a weighted average for each grid node. The weight assigned to the input data depends on the distance between the node considered and this value. The farther away from a given point, the smaller the influence of that point will be (inverse distance allocation). The allocator is exclusively used for distributed parameters. It is best used whenever moderately large amounts of point values are available. (External equivalent XInvDist).

C.2.9 TinInterpol
This allocator interpolates values given as point-data. The interpolation uses a Triangular Irregular Network, which is build from the given points. This network has to be larger than the grid; e.g. the data set has to contain point values outside the model's boundary. After linear interpolation, within the triangles of the TIN, values are assigned to the nodes of the grid. The allocator is best used whenever large amounts of point values are available. (External equivalent Xtin).

C.3 External Allocators

C.3.1 XInvDist

The allocator interpolates values that are given as point-data. The interpolation is weighted by the inverse distance between the node considered and the data point. The farther away from a given point, the smaller the influence of that point will be. The external allocator is used for distributed parameters or for boundary parameters.

The following options are available, to be entered after starting the program:

-q Quadrant Search.
In every quadrant a selected number of points (by default 4) are used for interpolation.

-n***X*** Set the maximum number of points.
The maximum number of points to be used in the interpolation is set to ***X***; by default the program uses the nearest 25 points for interpolation.

-r***X*** Set the maximum Search Radius.
The maximum search radius is set to ***X*** (in user units); for each node only points that are within this distance are used in the interpolation, points that are outside this radius are neglected.

-b Interpolation of values at the boundary-nodes only.

The allocator is best used whenever moderately large amounts of point values are available. Depending on the amount of point values and the options used some smoothing of results may occur.

Command line call:
xInvDist <set-dir> <grid-dir> <config-file>

Example:
xInvDist C:\model\cal C:\model\grid RP1.cfg

C.3.2 XTin

This allocator interpolates between values given as point-data. The interpolation uses a Triangular Irregular Network, which is build from the given points. This network has to be larger than the grid; e.g. the data set has to contain point values outside the model's boundary. After linear interpolation, within the triangles of the TIN, values are assigned to the nodes of the grid. The allocator is best used whenever large amounts of point values are available.

Two options are available that can be entered after starting the program:

-n Natural Neighbour Search:
The program will use non-linear interpolation, such that the first derivative in every point will be continuous.

-b Interpolation of values at the boundary-nodes only.

Command line call:
xTIN <set-dir> <grid-dir> <config-file>

Example:
xTIN C:\model\cal C:\model\grid RP1.cfg

C.3.3 xParAdo, xParBou, xParRiv, xSrcParAdo,xConst

Command line call:
xParAdo <set-dir> <grid-dir> <config-file>

Example:
xParAdo C:\model\cal C:\model\grid RP1.cfg

C.3.4 Kriging

This allocator interpolates values defined by polygons, lines and points. The program computes a variogram from the given point values and interpolates the values using this variogram as a weighing factor. The program is best used whenever data are rather scarce, irregularly scattered or if inhomogeneities (geological

fracture zones) are present.

Kriging contains some usefull functions shown below.

- interpolate with points and lines

- inside polygons constant value or interpolate in polygons when polygonvalue = -999

- maximum of 300000 points in grid or map

- nuggetvar.=0: determine variogram without nugget variance

Command line call:
kriging <gridfile> <par-file> <ung-file> <setname> <defaultval.> <outputfile> <nuggetvar. 0 or 1>

Example:
kriging ..\grid\grid.teo ..\initial\test.par ..\calib\test.ung test 100 test.ado 0

C.3.5 Arpadi-family

Arpadi is a specific allocator, included in the **Triwaco-**package as an external program. Sub areas (polygons) and point values define the value of the distributed parameter. Within each sub area, the parameter is assumed to be a constant. At the start the program checks whether or not the grid nodes are within one of the sub areas. These nodes are assigned the value of the corresponding polygon. If there are no point values defined, grid nodes outside the sub areas are assigned the default value (defined in the ***'Parameter Info'*** screen)

The numbering of the IDs in the *.par file of a parameter to be allocated with ***Arpadi*** may contain gaps; sequential numbering is also not needed (the numbering may be random). However, the parameter file has to end with a carriage return. If not, the program generates the error message: "ID from .UNG file missing in . PAR file".

Using Arpadi with ***'user defined parameters'*** one should consider the following restrictions in name giving.

A. The parameter name should not exceed 20 (alphanumeric) characters.

B. The total string of the parameter's *.ung and *.par files, including the total path, should not exceed a total of 60 characters.

> Thus the following 64-character string
> "**C:\directory\subdirectory\another\subdirectory\parametername.ung**"
> will be read by the program as the next 60-character string, neglecting the file's extension:
> "**C:\directory\subdirectory\another\subdirectory\parametername**".

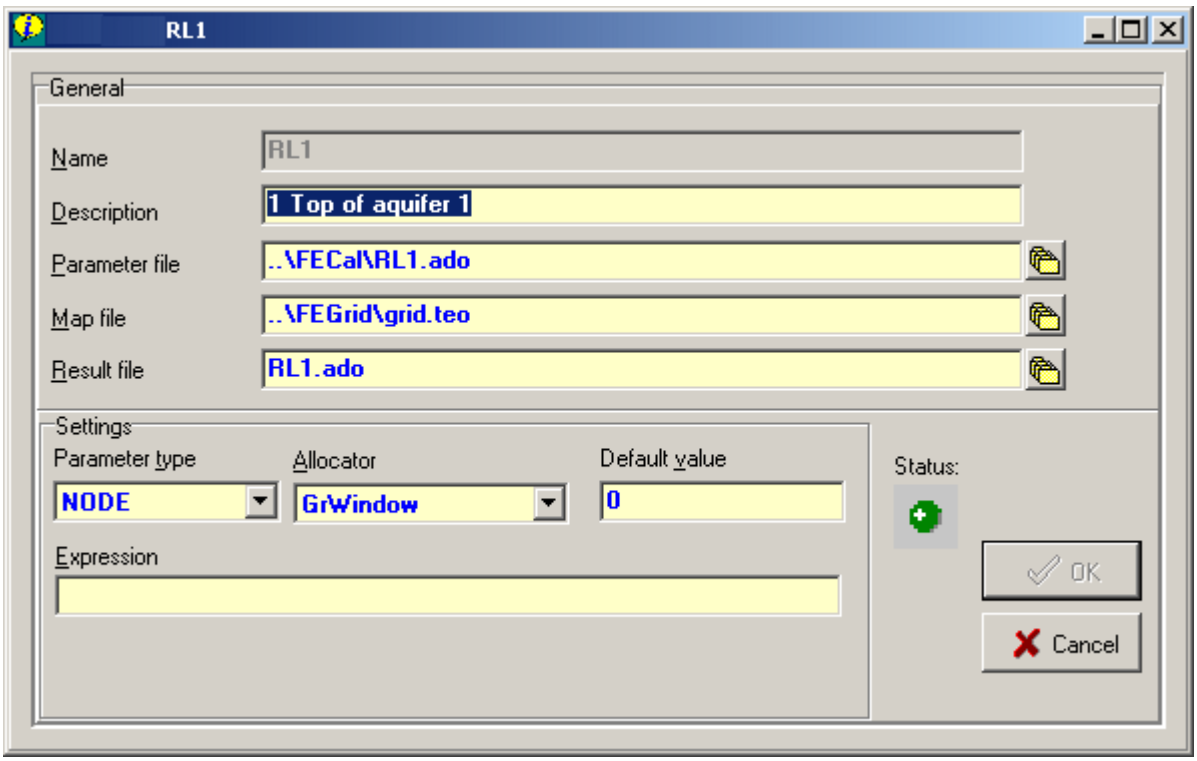
C. The first two characters should not correspond with the two first characters of a Boundary, River or Source parameter. In that case the program generates an error message stating that the data-type is not correct. The following two-character strings should be avoided in naming a user defined parameter: **'IB', 'BH', 'BA'** and **'BB'** (exclusively used for boundary parameters); **'RA', 'HR', 'RW', 'CI', 'CD', 'RD', 'RQ'** and **'RC'** (exclusively used for river parameters); and **'IS', 'SH', 'SQ'** and **'SN'** (exclusively used for source parameters).

Command line call:
Arpadi <set-dir> <grid-dir> <config-file>

Example:
ArpLin C:\model\cal C:\model\grid RP1.cfg

C.3.6 GrWindow and SGrWindow

Usage of (S)grWindow requires special attention in the '***parameter info window***'. An example is shown below. In the parameter file the result file from the large model is defined. As map file the grid file of the larger model. Usage is the same for SgrWindow.


RL1
General
Name
RL1
Description
1 Top of aquifer 1
Parameter file
..\FECal\RL1.ado
Map file
..\FEGrid\grid.teo
Result file
RL1.ado
Settings
Parameter type
NODE
Allocator
GrWindow
Default value
0
Expression
Status:
OK
Cancel

C.4 Expression

C.4.1 Operators

The expression allocator assigns the result of an expression to the nodes. The expression has to be defined in the ***'Parameter Info'*** dialog box. The allocator may be used for all types of parameters, provided the parameters referred to in the expression are of the same type. For parameters of different type a conversion of the type may be needed, for which the expression allocator provides a number of conversion-functions.

An expression may contain set-names, numbers, functions, factors and operators. Three types of operators may be distinguished: mathematical operators, relational operators and logical operators.

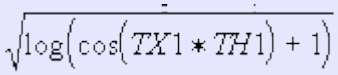
Definition	Description	
Set-names	Parameter names as defined in **Triwaco**, consisting of a combination of alphanumeric characters. The parameter may be preceded by the name of one of the project's data sets and a $-sign: e.g., cal$TX1	
Numbers	integer and real numbers: e.g., 15, -0.456	
Factors	Consist of numbers, expressions, functions or identifiers.	
Mathematical operators	+, -, * and /	
Relational operators	>, ≥ **(>=)**, = **(==)**, ≤ **(<=)** and <	
Logical operators	'AND' ('**&&**'), 'OR' ('**||**') and 'NOT' ('**=!**') and 'IF' 'THEN' ('**?**') and 'ELSE' ('**:**')	
Functions	(simple) mathematical functions:	
	abs(x)	Returns the absolute value of '***x***'
	atan(y,x)	Returns the arc tangent of ('***y/x***')
	BND(x)	Returns the value of '***x***' at boundary nodes
	cos(x)	Returns the cosine of '***x***'
	deg(x)	Converts radians ('***x***') to degrees
	exp(x)	Returns the value of ***e*** raised to the power '***x***'
	IF(x,y,z)	Evaluates the logical expression: IF ('***x***') THEN ('***y***') ELSE ('***z***') Equivalent to the expression: ('***x***')?('***y***'):('***z***')
	ln(x)	Returns the natural logarithm of '***x***'
	log(x)	Returns the 10 log of '***x***'
	max(x,y)	Returns the largest value of '***x***' and '***y***'
	min(x,y)	Returns the smallest value of '***x***' and '***y***'
	NODE(x)	Returns the value of '***x***' at **all Nodes**; if the value of '***x***' does not exist at a Node a zero value (**0**) is assumed
	rad(x)	Converts degrees ('***x***') to radians
	RIV(x)	Returns the value of '***x***' at river nodes
	sign(x)	Returns the sign of '***x***' (-1, 0 or +1)
	sin(x)	Returns the sine of '***x***'
	sqr(x)	Returns the square of '***x***'
	sqrt(x)	Returns the square root of '***x***'
	SRC(x)	Returns the value of '***x***' at source nodes
	tan(x)	Returns the tangent of '***x***'

Important note: The setname or data set name should NOT contain an underscore (data_set$set_name).

C.4.2 Examples of expressions

In the following table examples of the more or less frequently used expressions are listed.

PHIT	adore block with values equal to those of the set with the matching set name: 'PHIT'
Result$PHI1	adore block with values equal to those of set 'PHI1' belonging to the data set with the name: 'result'
12	adore block with the constant value 12
PHI1-PHIT	adore block with values equal to (PHI1 - PHIT), being the difference of the adore blocks with set names 'PHI1' and 'PHIT' respectively
QRCH>0	Boolean adore block containing **integer** values: equal to **1** where QRCH > 0 and equal to **0** where QRCH <= 0
(PHI1-PHIT) * (QRCH>0 && QKW1>0)	Real adore block containing values equal to **0** where QRCH <= 0 or QKW1 <= 0 and to (**PHI1-PHIT**) where both QRCH > 0 and QKW1 > 0
(RL1>TH1)?RL1:(TH1 + 0.01)	Real adore block containing values equal to **RL1** where RL1 > TH1 and to (**TH1+0.01**) where RL1 <= TH1
IF(RL1>TH1,RL1,TH1+0.01)	Real adore block containing values equal to **RL1** where RL1 > TH1 and to (**TH1+0.01**) where RL1 <= TH1

sqrt(log(cos(TX1*TH1)+1)	adore block that contains values equal to the results after evaluating the expression: 
QRI1/AREA	Specific river flux in m/d (river flux divided by node influence area)
MIN(PHIT,RP13)	Minimum value of **PHIT** and **RP13**: cut off **PHIT** at surface level
PHIT > RP13 ? RP13 : PHIT	Same as above
IF(PHIT>RP13, RP13, PHIT)	Same as above

Note:
Using Boolean expressions the result set will contain **integer** values if the expression starts with the Boolean expression and will contain **real** values if the Boolean expression is preceded with a (real) value or another expression.

Thus: (PHI1-PHIT) * (QRCH>0 && QKW1>0) results in a **real** Adore set and
(QRCH>0 && QKW1>0) * (PHI1-PHIT) results in an **integer** Adore set.

C.4.3 Complete expression syntax

The following table summarizes the complete expression syntax.

expression =	logical_expression
logical_expression =	relational_expression
	relational_expression '**&&**' relational_expression
	relational_expression '**||**' relational_expression
relational_expression =	additive_expression
	additive_expression '**<**' additive_expression
	additive_expression '**>**' additive_expression
	additive_expression '**<=**' additive_expression
	additive_expression '**>=**' additive_expression
	additive_expression '**==**' additive_expression
	additive_expression '**=!**' additive_expression
additive_expression =	multiplicative_expression
	multiplicative_expression '**+**' multiplicative_expression
	multiplicative_expression '**-**' multiplicative_expression
multiplicative_expression =	term
	term '*****' term
	term '**/**' term
term =	typed_factor
	typed_factor '**^**' typed_factor
	[typed_factor '**^**' typed_factor]...
typed_factor =	factor
	'-' factor
	+' factor
	'!' factor
factor =	number
	'('expression')'
	identifier
	function(expression)
identifier =	alphanumeric string
	quoted alphanumeric string
function =	abs(..), min(..,..), max(..,..) and sign(..)
	log(..), ln(..) and exp(..)
	sqr(..) and sqrt(..)
	sin(..), cos(..), tan(..), atan(..,..), deg(..) and rad(..)
	IF(..,..,..)