# Groundwater Data Utilities

## Part B: Program Descriptions

**Watermark Computing**

**July, 2005**

**PREFACE**

Part B of the manual for the Groundwater Data Utilities contains individual program descriptions. Programs are discussed in alphabetical order.

Refer to Part A of the Groundwater Data Utilities manual for an overview of the utilities, a description of the file types used by them and for a discussion of a number of common groundwater data processing tasks that can be accomplished using them.

Part B begins with a table listing all current utilities. Also listed are the programs belonging to the PEST MODFLOW/MT3D Utilities suite (largely superseded by the Groundwater Data Utilities).

## ALPHABETICAL LISTING OF GROUNDWATER DATA UTILITIES

| | |
|---|---|
| Adjobs | Adjusts observation weights for different observation groups in a PEST control file according to user-defined formulae. |
| arrayobs | Facilitates the introduction of model outputs comprised of MODFLOW/MT3D-compatible real arrays into a PEST parameter estimation process. |
| bud2hyd | Extracts flow data from a MODFLOW unformatted cell-by-cell flow term file. Rewrites this data in a form suitable for plotting against time. |
| bud2smp | Extracts flow data from a MODLFOW unformatted cell-by-cell flow term file. Rewrites this data in bore sample file format. |
| fac2fem | Uses ppk2fac-generated kriging factors to produce a MicroFEM input file on the basis of spatial interpolation from a set of pilot points. |
| fac2mf2k | Modifies an existing set of MODFLOW-2000 input files, replacing parameter cited in that file with pilot-point-based parameters (often a first step in pilot-point-based model calibration). |
| fac2real | Uses ppk2fac-generated kriging factors to produce a MODFLOW-compatible real array on the basis of spatial interpolation from a set of pilot points. |
| fem2smp | Converts MicroFEM output to bore sample file format. |
| fieldgen | Generates a stochastic field in each zone of a model domain using the sequential Gaussian simulation method. |
| Genreg | Inserts prior information pertaining to many different types of regularisation into an existing PEST control file. |
| grid2arc | Writes ARCINFO generate files of the active part of the finite-difference grid as defined by a user-supplied integer array. |
| grid2bln | Writes a SURFER blanking file of the active part of the finite-difference grid as defined by a user-supplied integer array. |
| grid2dxf | Writes a DXF file of the active part of the finite-difference grid as defined by a user-supplied integer array. |
| grid2pt | Tabulates the coordinates of the cell centres of the finite-difference grid within an active window defined by a user-supplied integer array. |
| int2mif | Generates MAPINFO MIF and MID files based on a MODFLOW/MT3D-compatible integer array. |
| int2real | Builds a MODFLOW/MT3D-compatible real array based on the contents of a MODFLOW/MT3D-compatible integer array. |
| Laydiff | Evaluates head value differences in different layers based on contents of a bore sample file, bore coordinates file and bore listing file. |

| | |
|---|---|
| many2one | Splits MODFLOW/MT3D-generated unformatted files comprised of multiple two-dimensional results arrays into individual formatted/unformatted files. |
| mod2obs | Interpolates model-generated data to the same times and locations as those cited in a user-supplied bore sample file; writes another bore sample file. |
| mod2smp | Interpolates the information contained in an unformatted MODFLOW/MT3D output file to a set of user-specified bores, rewriting the bore-specific data as a bore sample file. |
| parm3d | Assists in the pilot-point parameterisation of a 3-d model domain where hydrogeologic units intersect grid layers. |
| pestprep | Automates construction of a PEST control file and PEST instruction file for a model comprised of MODFLOW and/or MT3D followed by MOD2OBS, or MODFLOW followed by BUD2SMP followed by SMP2SMP. |
| pmp2info | Builds a bore information file from a bore pumping file, the former containing cumulative pumped volumes between two user-specified dates for a user-supplied list of bores. |
| pmpchek | Checks the integrity of the data contained in a bore pumping file. |
| ppcov | Builds a covariance matrix pertaining to pilot point parameters based on one or a number of geostatistical structures. |
| ppk2fac | Calculates kriging factors for use in spatial interpolation from a set of pilot points to model grid cell centres. |
| ppk2facf | Calculates kriging factors for use in spatial interpolation from a set of pilot points to the nodes of a MicroFEM finite element mesh. |
| ppk2fac1 | Identical to ppk2fac except for the fact that the regularisation data file which it writes is suitable for the use of ppkreg1. |
| ppkreg | Adds a "prior information" and "regularisation" section to a PEST control file where parameterisation is based on pilot points. |
| ppkreg1 | Similar to ppkreg but more powerful in that it facilitates the use of both "smoothness regularisation" (same as ppkreg) and "preferred value regularisation". |
| pt2array | Builds a MODFLOW-compatible real array; the value assigned to each array element is calculated from information pertaining to points lying within the respective element. |
| ptingrid | Locates the finite-difference cells in which arbitrary, user-supplied points lie; optionally provides the value of an integer or real array element pertaining to the cell containing each such point. |
| qdig2dxf | Translates the output of the shareware digitizing program, QDIGIT, into DXF format. |

| | |
|---|---|
| qdig2xyz | Translates the "contours" output of QDIGIT to an "*xyz*" data file. |
| real2int | Builds a MODFLOW/MT3D-compatible integer array based on the contents of a MODFLOW/MT3D-compatible real array. |
| real2mif | Generates MAPINFO MIF and MID files based on a MODFLOW/MT3D-compatible real array. |
| real2srf | Translates a MODFLOW/MT3D-compatible real array into a SURFER grid file. |
| real2tab | Translates a MODFLOW/MT3D-compatible real array into three-column real array table format. |
| reparray | "Pastes" a MODFLOW or MT3D compatible real array into an existing MODFLOW or MT3D input file. |
| rotbln | Rotates a SURFER blanking file about the top left corner of a finite-difference grid so that the component elements of the file can be overlain over the grid when the latter has been rotated such that its row direction is oriented directly east. |
| rotdat | Rotates a data file about the top left corner of a finite-difference grid so that the component elements of the file can be overlain over the grid when the latter has been rotated such that its row direction is oriented directly east. |
| rotdxf | Rotates a DXF file about the top left corner of a finite-difference grid so that the component elements of the file can be overlain over the grid when the latter has been rotated such that its row direction is oriented directly east. |
| section | Interpolates the data contained in multiple MODFLOW-compatible real arrays to an arbitrary transect line through all or part of the finite-difference grid. |
| smp2dat | Extracts information from a bore sample file for a user-supplied list of bores at times corresponding to model output times, generating a bore data file from which PEST input files can be constructed using the PEST MODFLOW/MT3D Utilities. |
| smp2hyd | Rewrites the contents of a bore sample file for a user-specified list of bores in a form suitable for plotting borehole data against time. |
| smp2info | Time-interpolates the information contained in a bore sample file to a user-specified date for a list of user-specified bores, thus writing a bore information file ready for access by commercial contouring software. |
| smp2pm1 | Re-writes the information in a bore sample file for user-selected bores as a PMWIN observation file. |
| smp2pm2 | Interpolates the information contained in a bore sample file to model output times, then writes the time-interpolated data as a PMWIN observation file. |

| smp2smp | Interpolates data contained within one bore sample file to the dates and times represented in another bore sample file. |
|---------|-------------------------------------------------------------------------------------------------------------------------|
| smpcal  | Calibrates one time-series dataset on the basis of another. |
| smpchek | Checks the integrity of a bore sample file. |
| srf2real | Re-writes a SURFER grid file as a MODFLOW-compatible real array. |
| tab2int | Generates a MODFLOW/MT3D-compatible integer array from an integer array stored within a GIS. |
| tab2real | Generates a MODFLOW/MT3D-compatible real array from a real array stored within a GIS. |
| tabconv | Translates between integer or real array table files using row/column identifier format and those using cell number identifier format. |
| twoarray | Combines two real arrays by addition, subtraction, multiplication, division and partial replacement. |
| vertreg | Adds "vertical regularisation" prior information to a PEST control file where parameterisation is based on pilot points. |
| zone2bln | Writes a SURFER "blanking" file of finite-difference grid zonation as defined by a user-supplied, MODFLOW-compatible integer array. |
| zone2dxf | Writes a DXF file of finite-difference grid zonation as defined by a user-supplied, MODFLOW-compatible integer array. |

**ALPHABETICAL LISTING OF PEST MODFLOW/MT3D UTILITIES**

| | |
|---|---|
| arrayrep | Replaces an array written to a MODFLOW/MT3D input file with an external, user-supplied MODFLOW/MT3D-compatible array. |
| compfile | Compares two files; identifies the location of the first difference between them. |
| modbore | Interpolates the data contained in unformatted MODFLOW-output head/drawdown files to the sites of user-specified bores. |
| modbore1 | An alternative version of this program for reading MODFLOW unformatted files generated by versions of MODFLOW compiled with certain FORTRAN compilers. |
| modgen | Generates a PEST control file based on the information contained in a bore data file and a parameter value file. |
| modins | Generates a PEST instruction file based on the contents of a bore data file and a MODBORE/MT3D output file for the current model. |
| mt3bore | Interpolates the data contained in unformatted MT3D-output concentration files to the sites of user-specified bores. |
| textrep | Replaces occurrences of user-defined text strings in a model input file with text strings (eg. parameter space identifiers) of the same length. |

## ALPHABETICAL LISTING OF PROGRAMS SUPPLIED WITH PEST

| | |
|---|---|
| inschek | Checks the integrity of a PEST instruction file; optionally reads a model output file using the instructions contained in the instruction file. |
| eigproc | Summarises sensitivity and eigenvalue information on potentially troublesome parameters. |
| jacwrit | Formats the Jacobian matrix for user-inspection. |
| jco2jco | Builds a Jacobian matrix file pertaining to a new PEST control file on the basis of that pertaining to an existing PEST control file. |
| jcotrans | Translates an unformatted Jacobian matrix file from old binary format to new (compressed) binary format. |
| par2par | A generalised, model-independent parameter pre-processor. |
| paramfix | Facilitates the fixing of parameters, and introduction of prior information pertaining to individual parameters, to an existing PEST control file. |
| parcalc | Used by PEST when undertaking SVD-assisted parameter estimation for calculation of base parameter values from super parameter values. |
| parrep | Re-builds a PEST control file on the basis of a new set of parameter values. |
| pest | Version of PEST in which PEST runs in one window and the model runs in another. |
| pestchek | Checks the integrity of an entire PEST input dataset. |
| pestgen | Generates a PEST control file based on the data contained in a parameter value file and an observation value file. |
| picalc | Used by PET when undertaking SVD-assisted parameter estimation for calculation of prior information equations expressed in terms of base parameters. |
| ppause | Pauses PEST execution. |
| ppest | Parallel PEST |
| pslave | Slave for Parallel PEST. |
| pstop | Stops PEST execution without a statistical printout. |
| pstopst | Stops PEST execution with a statistical printout. |
| punpause | Unpauses PEST execution. |
| sensan | Sensitivity Analyser – allows a user to undertake many model runs using PEST model communications protocol. |
| senschek | Checks the integrity of a SENSAN input dataset. |

| svdaprep | Automates PEST setup for SVD-assisted parameter estimation. |
|---|---|
| tempchek | Checks the integrity of a PEST template file; optionally constructs a model input file based on the template file and the data contained in a parameter value file. |
| wtfactor | Multiplies observation weights pertaining to a selected group by a user-specified factor. |

# ADJOBS

**Function of ADJOBS**

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

- introduce new observation groups on the basis of observation names, and

- calculate observation weights on the basis of observation values; different formulae can be used for weights calculation for different observation groups.

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

**Using ADJOBS**

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

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

First:-

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

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

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

Division into subgroups takes place on the basis of observation names. Thus if different observation types have different name types, these different observation types can be easily placed into different groups. For example if discharge observations are named *dis001, dis002, dis003,* etc. and pollutant concentrations are labelled *conc001, conc002, conc003,* etc, and both of these observation types presently belong to the same group, they can be separated on the basis of the first three letters of each of their observation names, with observations beginning with the letters "*dis*" being assigned to one group and those beginning with the letters "*con*" being assigned to another group. To achieve this, answer the following prompt with the number "3":-

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

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

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

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

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

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

Then, if "y",

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

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

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

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

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

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

**Uses of ADJOBS**

As has already been mentioned, ADJOBS is particularly useful in PEST preprocessing where PEST is to be used in the estimation of parameters for a model which generates one or a number of lengthy time series. Because there are often many measurements comprising an observation dataset in this context, preparation for a

PEST run requires software assistance. Such assistance is available through PESTPREP, whose use is documented elsewhere in this manual. However PESTPREP does not allow automatic generation of observation groups on the basis of observation types. Nevertheless, it does allow observations of different types to be named differently. Group assignment can then be carried out using ADJOBS.

ADJOBS can also be useful where different types of data are used simultaneously in model calibration. By assigned each observation type to a different observation group, it is possible to rapidly adjust the relative weighting assigned to each such group in the calibration process through the use of ADJOBS.

**See Also**

See also MOD2OBS, PESTPREP and SMP2SMP.

# ARRAYOBS

**Function of ARRAYOBS**

The purpose of ARRAYOBS is to facilitate the introduction of a new dataset (of a certain type) into an existing inversion process. ARRAYOBS modifies an existing PEST control file and builds a new instruction file to allow inclusion of this data and the reading of model outputs corresponding to this data.

Use of ARRAYOBS is predicated on the assumption that two "three-column real array" files exist. The format of these files is documented in Part A of the manual of the Groundwater Data Utilities; see also the description of program REAL2TAB later in this document. Such a file holds the same data as a MODFLOW/MT3D-compatible real array; however the data is arranged in columns so that it is more easily readable by other software (such as a GIS or spreadsheet package). One of these two files must contain "the observation dataset" – that is, the data that will be recorded in the PEST control file; this is the data that model outputs must try to match. The other three-column real array file must have been produced by the model. While ARRAYOBS does not actually use the data contained in this file, it does check that the same cells are represented in this file as are represented in the observation three-column real array file. It also generates an instruction set to read this file. Thus ARRAYOBS acts as a PEST pre-processor for an inversion process in which one of the model output files is a three-column real array table file. After each model run, PEST will read this file, attempting to match the numbers in this file to the corresponding numbers in the observation three-column real array table file.

Note that a real array table file need not contain data pertaining to every cell in the model grid. Program TAB2REAL (which generates such a file from a real array file) allows use of an integer array for selection of array elements for representation in the table file.

**Using ARRAYOBS**

A settings file `settings.fig` must be present in the directory from which ARRAYOBS is run. See Part A of the manual to the Groundwater Data Utilities for further details.

ARRAYOBS commences execution with the prompt:

```
 Enter name of grid specification file:
```

to which you should respond by entering the appropriate filename. If a default filename for the grid specification file has been read from a filename file (`files.fig`) resident in the current directory, that filename will appear with the above prompt. It can be accepted through pressing the <Enter> key or rejected by supplying the correct filename. The grid specification file contains the dimensions and geometry of the finite-difference model grid. Its specifications are explained in Part A of the manual of the Groundwater Data Utilities.

Note that, as is the case for other members of the Groundwater Data Utilities, responding to any prompt with an "e" (for "escape") takes you back to the previous prompt. In this way mistakes can be quickly corrected.

ARRAYOBS next prompts for the name of an existing PEST control file:

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

and then for the names of two three-column real array table files:-

```
 Enter name of observation real array table file:
 Enter name of model real array table file:
```

Each of these files will probably have been written by program REAL2TAB (ARRAYOBS assumes use of exact REAL2TAB format when generating instructions to read the second of these files). However it is assumed that the first file contains "observation data" and that the second file is an example of a model output file to be generated repeatedly during the forthcoming parameter estimation process; part of the aim of this process will be to match a model output real array to an observation real array. As is the protocol for a real array table file, not all real array elements need to be represented in this file. However it is essential that the two files named in response to the above prompts contain data pertaining to exactly the same cells of the finite difference grid and that the array elements are listed in the same order. (This is easily ensured by using the same "window integer array" when generating these files using REAL2TAB.) Array data extracted from the observation real array table file will be written to the "observation data" section of the PEST control file. An instruction file will be generated to read the model real array table file. However prior to undertaking any of these activities, ARRAYOBS will ensure that the two files named in response to the above prompts are matched element for element.

When adding data to the PEST control file, and when writing instructions to read the model real array table file, ARRAYOBS provides names for the observations pertaining to real array elements. The naming protocol is "*pr_row_col*", where "pr" is a two character prefix, *row* is the cell row number, and *col* is the cell column number (these are both supplied in all three-column real array table files). The one or two character prefix is supplied by the user in response to the prompt:-

```
 Enter prefix for new observation names (two characters or less):
```

ARRAYOBS's next two prompts are:-

```
 Enter weight to assign to new observations:
 Enter group name for new observations:
```

As is explained in the PEST manual, each observation must be given a weight and must be assigned to a group. For observations assigned to the observation group *regul*, user-supplied weights are multiplied by a PEST-calculated "weight factor" (which is re-calculated during every optimisation iteration) to ensure that the pertinent observations feature in the inversion process as much as is required to enforce the regularisation constraints which they embody without, at the same time, detracting from PEST's ability to achieve a desired level of fit between model outcomes and field data.

If the user wishes that a more complex weighting strategy than that of a single uniform weight be applied to observations introduced to the inversion process by ARRAYOBS, he/she can use program ADJOBS from the Groundwater Data Utilities to adjust these weights. Note that, for observations belonging to the observation group *regul*, relatively of weights is preserved as weights are adjusted through the regularised inversion process.

ARRAYOBS next prompts for the name of the new PEST control file which it must write:-

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

Enter an appropriate name (an extension of "*.pst*" is mandatory).

ARRAYOBS next issues a special sequence of prompts if the existing PEST control file is not set up to run in regularisation mode, and if the name of the new observation group (as supplied by the user in response to the pertinent one of the above prompts) is "regul". Inferring that the user will want to be running PEST in regularisation mode, ARRAYOBS offers to add a "regularisation" section to the end of the new PEST control file which it is about to write. It asks:-

```
 Use regularisation mode for new PEST control file?  [y/n]:
```

If the response to this prompt is in the affirmative, ARRAYOBS prompts for the names of two regularisation control variables:-

```
 Enter value for PHIMLIM:
 Enter value for FRACPHIM:
```

It supplies default values for the other variables appearing in the "regularisation" section of the PEST control file which it generates; these can be easily altered by the user by direct editing of this file if desired. (Note that a value of zero for FRACPHIM is normally suitable; however be prepared to raise this value if PEST appears to be buffeted by the winds of numerical instability.)

Finally ARRAYOBS prompts for the name of the instruction file which will contain the instructions which PEST will use to read the model-generated real array table file:-

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

ARRAYOBS then writes the new instruction and PEST control files. In writing the new PEST control file, ARRAYOBS modifies the existing PEST control file in the following ways:-

1. It adds observations (based on the observation three-column real array file) to the "observation data" section of the PEST control file.

2. It increases the value of NOBS (number of observations) in the "control data" section of the PEST control file accordingly.

3. If the requested name for the observation group to which the new observations are assigned is not featured in the original PEST control file, ARRAYOBS

adds the name of this new group to the "observation groups" section of the PEST control file. It increments the control variable NOBSGP accordingly.

4. It records the names of the new instruction file and the corresponding model output file in the "model input/output" section of the new PEST control file. It also increments the NINSFLE variable in the "control data" section of the PEST control file.

5. If requested, it alters the PESTMODE control variable to "regularisation" and adds a "regularisation" section to the end of the new PEST control file.

**Uses of ARRAYOBS**

ARRAYOBS is particularly useful when preparing for a PEST run in regularisation mode where the regularisation condition is one of minimizing the discrepancy between a model output array and a "preferred system condition" array. The latter may be uniform, piecewise uniform, embody a data trend, contain a stochastic field, or represent the "preferred system condition" in any other way that is useful in the current modelling context. In all cases, when run in regularisation mode, PEST will be asked to adjust model parameters until the fit between model outputs and field measurements is reduced to a user-specified level (PEST variable PHIMLIM), but to do this in such a way that the selected model output array is as close as possible to the preferred system condition array. This model output array will often be the model hydraulic conductivity array (or the log of this array) calculated through "warping" of the "preferred system condition" array through use of a multiplier array. The latter may have been generated through interpolation between pilot points whose values are estimated by PEST through the inversion process. Alternatively, the model output array may be the multiplier array itself, this being matched to an observation array comprised entirely of ones during the inversion process (or zeros if the log of the multiplier array is used as the model output file). Whatever strategy is employed, the regularisation process used by PEST ensures numerical stability at the same time as it ensures minimum departure from the "preferred system state". If a series of such states is generated on the basis of geostatistical knowledge of an area, a series of calibrated models can be produced instead of just one. By using all of these models when making a prediction, the uncertainty pertaining to this prediction can be analysed.

**See Also**

See also ADJOBS, PESTPREP and REAL2TAB.

# BUD2HYD

**Function of BUD2HYD**

Program BUD2HYD reads an unformatted cell-by-cell flow term file written in COMPACT form by MODFLOW96 and later versions of MODFLOW. It extracts cumulative flow rates within each of a number of user-defined zones within the model domain for all times represented in the cell-by-cell flow term file. It then records these flow rates on its output file in a format that is readily acceptable to most commercial plotting packages. Thus flow rates within different parts of the model domain can be plotted against time.

**Using BUD2HYD**

Like many of the Groundwater Data Utilities, immediately upon commencement of execution BUD2HYD looks for a file named `settings.fig` in the subdirectory from which it was invoked. If this file is not found, BUD2HYD terminates execution with an appropriate error message. As is explained in Section 2.19 of Part A of this manual, the contents of file `settings.fig` inform those utilities that read it of the format to use in representing dates.

BUD2HYD's first prompt is:

 Enter name of grid specification file:

If a filename file (`files.fig`) residing within the working subdirectory contains the name of a grid specification file, that name will appear as part of the above prompt. The user can accept it simply by pressing the <Enter> key; alternatively, another filename can be supplied instead. BUD2HYD obtains the grid dimensions for the current model by reading the first line of the grid specification file. It obtains the number of layers in the current model from the user's response to the prompt:-

 How many layers in the model?

BUD2HYD next prompts for the name of the MODFLOW unformatted cell-by-cell flow term file which it must read:-

 Enter name of MODFLOW unformatted BUDGET output file:

It then asks:-

 Is this a MODFLOW88 or MODFLOW96 budget file  [8/9]?

If the user responds by typing "8", indicating that the file was produced by MODFLOW88, BUD2HYD terminates execution with an appropriate error message, for BUD2HYD can only process cell-by-cell flow term files produced by MODFLOW96 or later versions of MODFLOW. Furthermore, BUD2HYD can only read such files if they are stored in COMPACT form. This is because files stored in this form contain timing information (lacking in other forms of flow term storage)

which is essential for the recording of flow data in a manner that allows plotting against elapsed simulation time. Hence if BUD2HYD discovers that the cell-by-cell flow term file whose name was provided above is not, in fact, stored in COMPACT form, it terminates execution with an appropriate error message. If your MODFLOW pre-processor does not provide an option for file storage in this manner this is not a problem, for a user can easily create a MODFLOW OUTPUT CONTROL file him/herself which directs MODFLOW to store files in COMPACT form. See the MODFLOW96 and MODFLOW2000 manuals for details.

BUD2HYD's next prompt is:-

```
 Enter text to identify flow type:-
```

Whenever it writes an array to its cell-by-cell flow term file, MODFLOW first records an array header. The header contains timing information as well as a 16-character identifier of the flow type represented in the following array. Some of these identifiers are set out in the table below.

| Package | Identifier |
|---------|------------|
| bcf/lpf/huf | STORAGE |
| bcf/lpf/huf | CONSTANT HEAD |
| bcf/lpf/huf | FLOW RIGHT FACE |
| bcf/lpf/huf | FLOW FRONT FACE |
| bcf/lpf/huf | FLOW LOWER FACE |
| drain | DRAINS |
| recharge | RECHARGE |
| river | RIVER LEACKAGE |
| well | WELLS |

**Some text identifiers contained in cell-by-cell flow term array headers.**

The user should supply at least part of an appropriate text identifier in response to the above prompt. If the user-supplied text occurs within the array identifier supplied with a particular array header, that array will be processed by BUD2HYD. Otherwise the array is ignored. Note that in normal BUD2HYD usage, only enough text needs to be provided in response to the above prompt to uniquely identify one particular flow type. If the user wishes to plot more than one flow term type against time, he/she should run BUD2HYD more than once, supplying a different text string in response to the above prompt on each occasion. (Note that, as is explained below, the user may ascertain the text identifiers pertaining to various MODFLOW packages by reading BUD2HYD's record file after BUD2HYD has finished execution.)

BUD2HYD next prompts:-

```
 Enter simulation starting date [dd/mm/yyyy]:
 Enter simulation starting time [hh:mm:ss]:
 Enter time units employed by model [y/d/h/m/s]:
```

(Note that date representation will be in the format "mm/dd/yyyy" instead of "dd/mm/yyyy" if this is appropriately denoted in the settings file `settings.fig`.) BUD2HYD requires the above information so that it can record the date and time pertaining to every flow rate on its output file, this information being useable in certain plotting circumstances.

Next the user is required to supply model zonation information. Zonation is described by a series of integer arrays, one for each layer. A zone is thus defined as the collection of model cells to which the same integer value is assigned. Such integer arrays can be created in most MODFLOW graphical user interfaces; see section 2.10 of Part A of this manual for further details. **Note that cells which are assigned an integer value of zero are ignored.** Thus the user can ascertain the flows within a small part of the model domain by supplying a set of integer arrays which are everywhere zero except within that part of the domain which is of interest. Prompts are:-

```
 Enter name of integer array file for layer 1:
 Enter name of integer array file for layer 2:
 etc.
```

Note that the same integer array file can be supplied for more than one layer if desired.

Next BUD2HYD prompts for the name of its principal output file:-

```
 Enter name for time-series output file:
```

Then:-

```
 Enter flow rate factor:
```

A flow rate factor different from unity may be required to convert MODFLOW-generated flow rates to more appropriate units. Flow rates recorded in the MODFLOW cell-by-cell output file pertain to the length and time units used by MODFLOW; a user may prefer to use different flow rates when presenting model outputs in graphical form. Flow rates calculated for each MODFLOW zone are multiplied by the flow rate factor supplied above before being written to the BUD2HYD output file.

Part of a typical BUD2HYD output file is shown below.

```
 Flow_type      Elapsed_Time   Date         Time        Flow_rate_1   Flow_rate_2
  DRAINS        5.250000E+00   06/01/1991   06:00:00     -2499.2446    -3234.2343
  DRAINS        1.575000E+01   16/01/1991   18:00:00     -3709.3015    -2893.6432
  DRAINS        2.625000E+01   27/01/1991   06:00:00     -4729.2451    -2742.4234
  DRAINS        3.675000E+01   06/02/1991   18:00:00     -5616.1382    -2342.4234
  DRAINS        4.725000E+01   17/02/1991   06:00:00     -6421.5688    -1932.5234
  DRAINS        5.775000E+01   27/02/1991   18:00:00     -7165.6807    -1743.4323
```

**Part of a BUD2HYD output file.**

The BUD2HYD output file is immediately amenable to plotting by most commercial plotting packages. The first column contains the text annotation pertaining to the flow type as recorded in MODFLOW array headers; note that if the user supplied insufficient text to discriminate between different header types, both flow types will be recorded; if so, this will be immediately apparent from an inspection of the BUD2HYD output file. Such a file will be of limited use for plotting purposes.

The second column of the BUD2HYD output file contains elapsed time in model units. This is normally plotted against the data residing in columns five and above. Columns three and four list the date and time corresponding to each row. Finally, columns five and above contain flow rates summed within each user-defined zone. Column headers indicate the zones to which the various columns pertain; header format is "Flow_rate_$n$" where $n$ is the integer array value defining a particular zone. Note that where applicable (eg. for the drain, recharge, river, etc packages), negative values represent model outflows whereas positive values represent model inflows.

Before writing its output file, BUD2SMP prompts:-

```
 Assign flows to beginning, middle or finish of time step?   [b/m/f]:
```

If the user types "f", the time associated with each flow term (ie. each row in the above table) will be the same as the time of model output (ie. the end of a particular model time step). However this may be inappropriate in many instances, as the user may consider that flow rates should be plotted at a time corresponding to the middle of each time step for which they were evaluated. Thus if the user types "m" in response to the above prompt, elapsed times (as well as dates) appearing in the BUD2SMP output file will correspond to the middle of each model time step for which the flow terms were accumulated. Similarly, by typing "b", flow terms can be assigned to the beginning of pertinent MODFLOW time steps.

BUD2YD's final prompt is:-

```
 Enter name for run record file:
```

Upon completion of execution BUD2HYD writes a record of every array encountered in the MODFLOW cell-by-cell flow term file which it has just read. This is a very useful function, for even if you do not wish to record data in time series format for later plotting, BUD2HYD can be used to list the contents of an unformatted MODFLOW budget file. As was mentioned above, text identifiers are also recorded in this file (under the heading "flow type"); this information can be of use in designating flow types in later BUD2HYD runs. Data that has been stored in plot-ready form by BUD2HYD is indicated by a "yes" in the final column of its run record file. Part of such a file is shown below.

```
Stress_period  Time_step  Elapsed_time  Flow_type   Flow_processed_by_BUD2HYD
    1             1          10.500       STORAGE              no
    1             1          10.500       CONSTANT HEAD        no
    1             1          10.500       FLOW RIGHT FACE      no
    1             1          10.500       FLOW FRONT FACE      no
    1             1          10.500       FLOW LOWER FACE      no
    1             1          10.500       WELLS                no
    1             1          10.500       DRAINS               yes
    1             1          10.500       RECHARGE             no
    2             1          21.000       STORAGE              no
    2             1          21.000       CONSTANT HEAD        no
    2             1          21.000       FLOW RIGHT FACE      no
    2             1          21.000       FLOW FRONT FACE      no
    2             1          21.000       FLOW LOWER FACE      no
    2             1          21.000       WELLS                no
    2             1          21.000       DRAINS               yes
    2             1          21.000       RECHARGE             no
    2             2          31.500       STORAGE              no
    2             2          31.500       CONSTANT HEAD        no
    2             2          31.500       FLOW RIGHT FACE      no
    2             2          31.500       FLOW FRONT FACE      no
    2             2          31.500       FLOW LOWER FACE      no
```

**Part of a BUD2HYD run record file.**

**Uses of BUD2HYD**

Uses of BUD2HYD are many. However a particularly important role is the tabulation of outflows to river and drain cells through the course of a model run. In the former case model-generated river outflows can then be compared with observed river baseflows for calibration purposes.

Where drain cells are used to simulate mining operations, the use of BUD2HYD in conjunction with MODFLOW presents a useful means of examining mine inflow as the disposition of mining operations (and hence MODFLOW drain cells) changes over time. The user simply builds an integer array that is zero everywhere except at cells that fall within the mined area; (such cells can all be assigned an integer array value of 1). Drain cell disposition can vary from stress period to stress period as mining progresses. By running BUD2HYD after MODFLOW, a record of mine inflow vs time in plot-ready format can be obtained.

As was mentioned above, BUD2HYD can be used to simply obtain a table of contents of a MODFLOW budget file. However the user is reminded that BUD2HYD will only read such files if they were generated in COMPACT form by MODFLOW96 or later versions of MODFLOW.

**See Also**

See also BUD2SMP, MANY2ONE and SMP2HYD.

# BUD2SMP

**Function of BUD2SMP**

BUD2SMP is very similar to BUD2HYD. However instead of producing a file ready for use by a commercial plotting package, it writes flow rates accumulated within model domain zones to a bore sample file. This file can be processed using SMP2HYD in order to produce time series files in plotting format if desired. However, given that this role is already performed by BUD2HYD, the most useful deployment of BUD2SMP is in conjunction with SMP2SMP to produce a bore sample file which is matched to an existing bore sample file based on field measurements. Through use of PESTPREP, PEST input files can then be automatically generated by which a groundwater model is calibrated against outflow/inflow data.

**Using BUD2SMP**

Because of its similarity to BUD2SMP, BUD2HYD will not be described in detail; the user is referred to the documentation of program BUD2HYD for operational principals. Only aspects of BUD2SMP usage which are different from that of BUD2HYD are described below.

At a certain stage of its execution BUD2SMP prompts:-

```
 Enter maximum number of output times:
```

Through its OUTPUT CONTROL input dataset, MODFLOW is directed to provide cell-by-cell flow term output at the end of certain time steps. In response to the above prompt, you should inform BUD2SMP of the total number of time steps for which there is such cell-by-cell output. If you are unsure, simply enter a number that is likely to exceed the number of output times; BUD2SMP uses this number solely to dimension internal arrays. If the number is too large, it does not matter (except if it is so large that BUD2SMP runs out of memory). If it is too small, BUD2SMP will inform you of this later in its processing and ask that you run it again, supplying a higher number for this parameter.

Note that BUD2SMP can only process cell-by-cell flow term files stored in COMPACT format by MODFLOW and later versions of MODFLOW.

Like BUD2HYD, BUD2SMP prompts for an integer array pertinent to each layer. The model domain is divided into zones on the basis of integers assigned to various cells within the model domain. For each time step at which cell-by-cell flow term data was accumulated, BUD2SMP calculates the total flow within each of the non-zero zones defined within the supplied integer arrays; however the zero-valued zone (if it exists) is ignored as far as flow term accumulation is concerned. BUD2SMP records accumulated flows in a bore sample file, writing one entry for each zone for each time step for which flow data was recorded by MODFLOW.

In writing a bore sample file, BUD2SMP needs to know the "bore identifier" to assign to each zone. So for each non-zero zone that it finds in the supplied integer arrays it prompts:-

```
 Enter identifier for flows in zone n
```

where *n* is a zone-defining number occurring within the integer arrays. Supply a name comprised of 10 characters or less; no two zones should be supplied with the same identifier.

Like BUD2HYD, BUD2SMP writes two files. One is a bore sample file, the other is a file recording the details of all arrays found in the cell-by-cell MODFLOW output file. If you wish to simply know what arrays are present in this file without necessarily creating a bore sample file, run BUD2SMP, supplying a flow identification text string which does not match any of the array identifiers generated by MODFLOW – see the documentation of BUD2HYD for more details.

Flow rate data can be referenced to the beginning, middle or end of the time step in which it was recorded; see the documentation for program BUD2HYD for more details.

**Uses of BUD2SMP**

BUD2SMP can be very usefully deployed in a composite model together with PEST in estimating parameters on the basis of model inflows and outflows. The composite model will be comprised of MODFLOW followed by BUD2SMP followed by SMP2SMP. The last of these programs will generate a bore sample file in which model-generated flows are temporally interpolated to the dates and times of measured flows. Input file preparation for the PEST run then becomes a trivial task through the use of PESTPREP.

**See Also**

See also BUD2HYD, PESTPREP and SMP2SMP.

# FAC2FEM

**Function of FAC2FEM**

FAC2FEM carries out spatial interpolation from a set of pilot points to the nodes of a MicroFEM finite element mesh using kriging factors calculated by PPK2FACF. It writes to two kinds of MicroFEM input files, viz. a "fem" file and a "sto" file (or any file that has the same format as the latter type of file). When undertaking pilot-points-based MicroFEM calibration using PEST, FAC2FEM must be run as part of a "composite model" (comprised of at least FAC2FEM and MicroFEM itself) under the control of PEST.

**Using FAC2FEM**

FAC2FEM commences execution with the prompts:-

```
Enter name of interpolation factor file:
Is this a formatted or unformatted file?
```

The interpolation factor file is written by PPK2FACF, which must be run before FAC2FEM in order to generate a set of kriging factors by which spatial interpolation from a set of pilot points to the MicroFEM finite element mesh can be carried out.

Next FAC2FEM prompts for the name of the pilot points file containing data for interpolation to the mesh:-

```
Enter name of pilot points file:
```

FAC2FEM normally supplies the name of a pilot points file with this prompt – the name of the same pilot points file as that used by PPK2FACF to generate the kriging factors (this filename is stored in the interpolation factor file). This default filename can be accepted by simply pressing the <Enter> key.

FAC2FEM then prompts for the upper and lower interpolation limits:-

```
Enter lower interpolation limit:
Enter upper interpolation limit:
```

If the value assigned to any node is above or below these limits, this value will be reduced or increased respectively in order to respect these limits. Care should be taken in supplying these limits. They can be useful when kriging is based on the Gaussian variogram; sometimes the range of kriged values can exceed that of the data values when using this variogram (which is why it is often best to use the exponential or spherical variogram instead). In general, when undertaking pilot-point-based parameterisation under the control of PEST, these limits should be set outside the PEST-imposed upper and lower parameter bounds. If the interpolation limits are narrower than the parameter bounds limits, the parameter estimation process will be compromised as node values interpolated from PEST-adjusted pilot point values are altered in order to respect the interpolation limits. In extreme cases, one or more pilot

point values can become insensitive, and hence unestimable, if node values around them are determined by the interpolation limits rather than by the values assigned to the pilot points. If in doubt set these limits wide and let PEST's parameter bounds functionality ensure that reasonable values are assigned to mesh nodes.

Next FAC2FEM prompts:-

```
Enter name of existing mesh property file:
```

Before using FAC2FEM you should make a copy of the MicroFEM input file which contains the parameters which you wish PEST to calculate. In assigning interpolated values to mesh nodes, MicroFEM will make a copy of this file, replacing existing property values listed in this file with those interpolated from the set of pilot point values contained in the pilot points file. The property file can be of the "fem" or "sto" type.

MicroFEM next prompts for the property type whose values it will replace with pilot-point-interpolated values. For the sake of generality, this property type is denoted by its number:-

```
Enter property number for replacement:
```

This number refers to the position that the property occupies on each pertinent line of the "fem" or "sto" file. In a "fem" file properties are grouped by layer. The vertical resistance for layer 1 is listed first, followed by the transmissivity of layer 1; then follow the vertical resistance and transmissivity respectively for layer 2; and so on for each layer. In a "sto" (or equivalent) file, only a single property type is represented, ordered by layer. Thus the storage coefficient of layer 1 is listed first, followed by the storage coefficient of layer 2, etc.

Next FAC2FEM prompts:-

```
Enter name for new mesh property file:
```

This is the name of the file that FAC2FEM will write. This file will be identical to the existing property file except for the fact that the identified property type will be replaced by property values calculated through pilot point interpolation. **It is very important that the extension of this file be in accordance with its type.** If the file has an extension of "fem", FAC2FEM will assume that the "existing mesh property file" whose name was provide above is also a "fem" file, and will read and re-write it accordingly. For any other extension, the simple "sto" type file structure is assumed.

Finally FAC2FEM prompts:-

```
Enter value for elements to which no interpolation takes place:
```

No interpolation will take place to nodes within zones for which kriging factors were not computed by PPK2FACF, or to nodes that are further away from any pilot points than the PPK2FACF search radius. See the documentation of PPK2FACF for further details.

Once it has been supplied with all of the above information, FAC2FEM writes the new mesh property file and terminates execution.

Note that if you wish to assign a single value to all nodes within a particular zone of the finite element mesh, simply assign a single pilot point to that zone.

**Uses of FAC2FEM**

FAC2FEM is used in the second stage of the spatial interpolation process from a set of pilot points to a MicroFEM finite-element mesh. The first stage of the interpolation process is undertaken using PPK2FACF which calculates kriging factors. When pilot points are assigned values by PEST (normally as part of a regularised calibration process), FAC2FEM is run prior to MicroFEM as part of a "composite model" encapsulated in a batch file. Responses to its prompts are placed in a "response file" to which FAC2FEM is directed through the "<" character on its command line.

Note that, in general, pilot-point-based calibration should not be undertaken without regularisation because of the large number of parameters that normally require estimation. As is explained elsewhere in this manual, the more pilot points that are used for spatial parameterisation, the less likely is the resulting parameter field to have a "blotchy" appearance, or to possess unusually high or low values within small, local, areas of the model domain.

**See Also**

See also PPK2FACF and PPKREG.

# FAC2MF2K

**Function of FAC2MF2K**

FAC2MF2K modifies an existing MODFLOW-2000 input dataset, replacing selected parameters within that dataset with parameters based on pilot points. Thus if a modeller's GUI does not support the use of pilot points in the parameterisation process, use of FAC2MF2K allows a modeller to use them anyway. "Kriging factors", by which hydraulic properties are calculated at MODFLOW cell centres from properties assigned to pilot points, are built into MODFLOW-2000 *multiplier* arrays; normally these kriging factors will have been computed by program PPK2FAC. The zonation integer array pertaining to the pilot points is added to the MODFLOW-2000 *zone* file. Optionally, unwanted *multiplier* and *zone* arrays can be removed from the existing MODFLOW-2000 input dataset.

Pilot points can be an extremely powerful device for accommodating spatial heterogeneity within a model domain. However the method works best when there is a superfluity of such points, and where model calibration is undertaken using nonlinear parameter estimation software. If enough pilot points are employed, the parameter estimator will be able to find locations within the model domain where hydraulic property variations need to be introduced in order to allow model outputs to match field data well. This process is best undertaken using PEST in preference to the nonlinear parameter estimation functionality of MODFLOW-2000, because PEST includes an extremely powerful mode of operation known as "regularisation mode". When operating in this mode, numerical problems caused by over-parameterisation are removed through the introduction of "regularisation constraints". Using the PPKREG utility such constraints can be introduced on the basis of a geostatistical characterisation of the study area. The effect of this type of regularisation is to limit the degree of spatial heterogeneity introduced to the model domain to that which is just sufficient to obtain a user-specified level of fit between model outputs and field data.

Once a pilot-point-based MODFLOW-2000 input dataset has been built with the help of FAC2MF2K, it is easily translated to a PEST input dataset using the MODFLOW2000-to-PEST translator, MF2PEST. Once in this format, it is a simple matter to modify the PEST control file to accommodate regularisation information using the PPKREG utility.

Where MODFLOW parameter estimation is carried out using PEST, it is necessary that a specially modified version of MODFLOW-2000 known as MF2KASP be used to conduct forward model runs. This version of MODFLOW-2000 retains all existing MODFLOW-2000 functionality. However additional functionality has been added that optimises its use in the pilot points context. In particular, interpolation from pilot point locations to the finite-difference grid can be undertaken on the basis of logs of parameter values rather than parameter values themselves. This interpolation strategy is required where a parameter's distribution is characterised by a log variogram rather than a variogram based on native parameter values; a log variogram is required to

characterise the transmissivity and/or hydraulic conductivity distribution prevailing in most study areas. See Section 5 of Part A of this manual for further details.

**Using FAC2MF2K**

*General*

Before using FAC2MF2K you should have built a MODFLOW model using your favourite Graphical User Interface. In those layers or stress periods, and for those property types, for which you intend to base parameterisation on pilot points, a simplistic parameterisation scheme should be adopted - probably based on one or two parameter zones. The parameters associated with these zones should be restricted only to those layers or stress periods for which pilot points will be introduced; they should appear in no other layers or stress periods. FAC2MF2K will replace parameters based on these zones with a parameterisation scheme based on pilot points. Thus these initial, zone-based, parameters will be completely removed from the MODFLOW-2000 input dataset, and a suite of new parameters introduced in their place. If you know the names of the *multiplier* and *zone* arrays pertaining to these parameters (these are resident in the MODFLOW-2000 *multiplier* and *zone* input files), and if these arrays pertain to no other parameter types, then these too can be removed from the MODFLOW-2000 input dataset. This introduces savings in MODFLOW's memory requirements. (Note that MODFLOW's memory requirements are normally far greater when using pilot-point-based parameterisation than when using zone-based parameters.)

As presently coded, FAC2MF2K can introduce pilot-point-based parameterisation only into the MODFLOW-2000 *lpf* and *rch* processes. In spite of this restriction, this should cover most practical requirements for the use of this parameterisation scheme. If you wish to use pilot points to characterise array-based parameters such as evapotranspiration, or if you wish to use pilot points in conjunction with the MODFLOW-2000 *huf* process, this can easily be accommodated. Simply use the FAC2REAL utility documented herein as part of a "composite model" (run before MODFLOW by PEST during the parameter estimation process) to build a MODFLOW-compatible array based on pilot points; then direct MODFLOW to read this array using its OPEN/CLOSE functionality for reading external array files, or use the REPARRAY utility to "inject" the array into the MODFLOW-2000 input dataset.

Before FAC2MF2K is run, PPK2FAC must have been run. The purpose of PPK2FAC is to build a "kriging factor file" on the basis of a pilot points file. The kriging factor files contains the (cell-specific) factors by which pilot point values are multiplied in order to conduct spatial interpolation from those points to the finite-difference grid. As these factors are independent of the hydraulic property values associated with the pilot points, they can be assigned to the pertinent elements of MODFLOW-2000 *multiplier* arrays. MODFLOW-2000 parameters to which the multipliers within these arrays are applied are the hydraulic property values pertaining to the pilot points.

When running PPK2FAC, different pilot points can be assigned to different zones within the model domain. Zonation is defined by an integer array. Factors contained within the kriging factor file are dependent to some extent on the disposition of zones

to which the different pilot points have been assigned. Like the factors themselves, the zonation array upon which calculation of the kriging factors is based must also be incorporated into the MODFLOW-2000 dataset. This array is added to the MODFLOW-2000 *zone* file.

When introducing the new pilot-point-based parameters into the MODFLOW-2000 input dataset, these parameters can be given the same names as the pilot points themselves. Alternatively a new name can be provided for each parameter comprised of a user-specified prefix (of up to two characters in length) followed by the name of the respective pilot point. The latter naming mechanism must be used where more than one set of parameters is introduced into the MODFLOW-2000 dataset based on the same set of pilot points. For example, one such set may represent hydraulic conductivity in a particular model layer, while another set may represent hydraulic conductivity pertaining to another model layer; or one set of pilot points may represent the hydraulic conductivity assigned to many model layers comprising the same hydrogeological unit, while the other set may represent the vertical anisotropy of that unit. Still another set may represent the unit's specific yield.

*Parameter Replacement File*

FAC2MF2K is provided with the details of the parameter replacement process which it must undertake through a "parameter replacement file". An example of such a file is shown below. Note that all entries in a parameter replacement file are case-insensitive; however keywords in the example shown below are capitalised to better exemplify the structure of this file. Note also that a blank line can occur anywhere within a parameter replacement file; also, a line beginning with the "#" character is ignored.

```
REPLACE  hk
  OUT hk_1
  OUT hk_2
  OUT hk_3
  OUT hk_5
  OUT hk_6
  NEWPARAMLAYS 1 2
  TRANSFORMTYPE log
  MINPARVAL 1e-2
  MAXPARVAL 1e3
  MININTERPVAL 1e-2
  MAXINTERPVAL 1e3
END REPLACE

REPLACE  hk
  OUT hk_8
  NEWPARAMLAYS 3 3
  NEWPARAMPREFIX "k_"
  TRANSFORMTYPE log
  MINPARVAL 1e-2
  MAXPARVAL 1e3
  MININTERPFILE lowlim.ref
  MAXINTERPFILE highlim.ref
END REPLACE


REMOVEZONE  zhk1
REMOVEZONE  zhk2
REMOVEMULT  hk1
REMOVEMULT  hk2
```

**A Parameter Replacement File**

A parameter replacement file must include at least one "replacement segment"; two are shown in the example above. A replacement segment begins with the string "REPLACE *partype*" where *partype* is a parameter type that can occur in the MODFLOW-2000 *lpf* or *rch* process. Legal parameter types are "hk", "hani", "vk", "vani", "ss", "sy", "vkcb" and "rch". A parameter segment must end with the words "END REPLACE".

A replacement segment is comprised of a set of lines, each of which must contain a keyword followed by the value assigned to the variable associated with the keyword. For all but one keyword, each line within a replacement segment must have two entries, ie. the keyword followed by the value pertaining to the keyword. However the line containing the NEWPARAMLAYS keyword (or NEWPARAMPERS keword for parameters of type "rch") must have 3 items (including the keyword itself). Some keywords are mandatory and must be cited within each replacement segment; however some are optional. Some keywords can be repeated many times; for others, only one incidence of the keyword can occur. Keywords can be supplied in any order.

The OUT keyword is used to identify parameters to be removed from the existing MODFLOW-2000 dataset; it can be used many times within the one replacement segment. Each parameter identified in this way is removed from both the *lpf* or *rch* input file, and from the MODFLOW-2000 *sensitivity* input file (if this file is present). It is very important to ensure when designing the model that this parameter is assigned only to those layers or stress periods to which pilot-point-based parameters will be assigned. If this is not the case, the removal of that parameter from layers or stress periods in which it is not replaced by pilot-point-based parameters will leave a gap in the parameterisation of the model. (FAC2MF2K will detect this condition if it occurs and issue an appropriate error message.) It is also important to ensure that a parameter supplied after the OUT keyword is of the type indicated in the header to the replacement segment. Once again, if this is not done FAC2MF2K will cease execution with an appropriate error message.

In the example shown above, the first replacement segment instructs FAC2MF2K to remove parameters *hk1*, *hk2*, *hk3*, *hk5* and *hk6* from the MODFLOW-2000 input dataset; all of these parameters are of type *hk*. The second replacement segment instructs FAC2MF2K to remove parameter *hk8*. Because of the NEWPARAMLAYS setting (see below) for each of these replacement segments, parameters *hk1*, *hk2*, *hk3*, *hk5* and *hk6* must be restricted to layers 1 and 2 of the model, whereas parameter *hk8* must be restricted to layer 3.

The NEWPARAMLAYS keyword can only be used in replacement blocks which pertain to *lpf* parameters. This keyword informs FAC2MF2K of the model layers to which pilot-point based parameters are to be introduced. A NEWPARAMLAYS keyword must be followed by two integers, the first specifying the upper layer and the second specifying the lower layer in a sequence of layers to which the pilot-point-based parameters are to be added; these integers can be the same if there is only one layer in the sequence. Note also that if the layers into which the pilot-point-based parameters are to be introduced are non-contiguous, then you can use more than one NEWPARAMLAYS keyword to specify different layer sequences.

For *rch* package parameters (of which "rch" is the only parameter type), the NEWPARAMPERS keyword must be used in place of the NEWPARAMLAYS keword. Its operation is identical to that described above for the NEWPARAMLAYS keyword, except that it pertains to stress periods rather than model layers.

The number of pilot-point-based parameters introduced into the MODFLOW-2000 dataset through each replacement block is equal to the number of pilot points occurring in the pilot points file whose name is supplied to program FAC2MF2K in response to an appropriate prompt (see below). The name assigned to each introduced parameter is that of the pilot point which it represents unless a NEWPARAMPREFIX keyword is present in a replacement segment. If present, this must be followed by a character string of two characters or less in length (including the null string ""), which will be appended to the front of the name of each pilot point to form the name of the related parameter. (Note that quotes are optional unless the null string is supplied.) Where there is more than one replacement segment in a parameter replacement file, each must cite a different parameter prefix. If this is not done then, because each

segment will then refer to the same set of pilot points, there will be a duplication of parameter names.

When using MODFLOW-2000, a parameter can be log-transformed in the *parameter* and *sensitivity* processes. If a PEST input dataset is then built from a MODFLOW input dataset using the MODFLOW2000-to-PEST translator MF2PEST, the transformation status of each parameter is carried through to the PEST inversion process. The parameter transformation status of pilot-point-based parameters introduced to the MODFLOW-2000 dataset through the parameter replacement file is set by the TRANSFORMTYPE keyword; appropriate entries following this keyword are "log" or "none". (For the types of parameters that are associated with the MODFLOW *lpf* process, especially hydraulic conductivity, TRANSFORMTYPE is best set to "log"; *rch* parameters are often more easily estimated if they are untransformed.)

*It is very important to note that if interpolation for a set of pilot points is based on a log-variogram, then parameters based on those pilot points must be log-transformed in the parameter estimation process.* If FAC2MF2K detects that this rule has been violated it will cease execution with an appropriate error message.

As is discussed in Section 5 of Part A of this manual, the PEST-compatible version of MODFLOW-2000 named MF2KASP provides special functionality for computing internal MODFLOW hydraulic property arrays from parameter values that makes this version of MODFLOW-2000 much more appropriate for the use of pilot-point-based parameterisation than the native MODFLOW-2000. In particular, summation and multiplication of parameter values to form arrays can take place on the basis of parameter value logarithms rather than on the basis of native parameter values (which is equivalent to kriging on the basis of a log variogram), and an upper and lower limit can be placed on array values calculated in this way. The latter functionality circumvents the occurrence of the wild and erratic interpolated values that can sometimes appear when using certain types of variogram. As will be discussed below, the variables which govern this aspect of MF2KASP's functionality must be supplied in a special "*asp* input file", a file which FAC2MF2K writes itself in the course of preparing the new MODFLOW-2000 input dataset. However, before doing this, it must know the upper and lower array bounds to use for introduced pilot-point-based parameters (it knows whether to undertake native or logarithmic interpolation based on the contents of the kriging factor file which it reads - see below).

Identical options are available for imposition of array bounds when using FAC2MF2K to those available through the use of FAC2REAL; ie. upper and lower bounds can be applied uniformly or on a cell-by-cell basis. If an upper bound is to apply uniformly over the whole layer to which a pilot-point-based parameterisation scheme is to be introduced (applicable only to that parameter type of course), then it can be supplied following the keyword MININTERPVAL. However if you wish to apply an interpolation lower limit on a cell-by-cell basis, then a MODFLOW-compatible array of minimum values can be read from a file named after the MININTERPFILE keyword. Note that this file must be a formatted file. Depending on the setting of the `colrow` variable in the settings file `settings.fig` (which must be present in the directory from which FAC2MF2K is run) the array contained

in this file may, or may not, be preceded by a "number of columns, number of rows" header.

Similar considerations apply to the setting of maximum interpolated array values. The pertinent variables are MAXINTERPVAL and MAXINTERPFILE. Note that because of their mutually exclusive roles, a MININTERPVAL and MININTERPFILE keyword cannot both occur within the same replacement segment; similarly, MAXINTERPVAL and MAXINTERPFILE are mutually exclusive.

The MINPARVAL and MAXPARVAL keywords have different roles from the MININTERPVAL and MAXINTERPVAL keywords. The latter relate to array values *after* interpolation from the set of pilot points to the MODFLOW grid has been carried out (individual array values can be larger or smaller than any of the pilot-point values on which they are based). However MINPARVAL and MAXPARVAL apply to the parameters themselves. When FAC2MF2K writes a new MODFLOW-2000 *sensitivity* file containing the new parameters, it transfers the values supplied for MINPARVAL and MAXPARVAL to the BL and BU variables respectively occurring in this file. If parameter estimation is undertaking using the MODFLOW-2000 inversion engine, these limits are used for reference purposes only. However if the MODFLOW-2000 dataset written by FAC2MF2K is translated to PEST format by the MODFLOW2000-to-PEST translator MF2PEST, then upper and lower parameter bounds supplied as MINPARVAL and MAXPARVAL are rigidly enforced.

Two additional keywords can (optionally) be supplied within a parameter replacement file; however they must not be used within a replacement segment. These are REMOVEZONE and REMOVEMULT. If, after removal of certain parameters as specified in one or a number of OUT keywords in one or a number of replacement segments, certain *zone* and *multiplier* arrays from the MODFLOW-2000 *multiplier* and *zone* input files are no longer needed, then these arrays can be removed from those files, thus reducing MODFLOW storage requirements when it is run to undertake a simulation or calibration task. Arrays within each of these files are specified by name; to remove an array, supply its name after the REMOVEZONE or REMOVEMULT keyword. Note that if a *multiplier* array in a *multiplier* file is specified by a FUNCTION, and if any of the arrays referred to in the FUNCTION have been removed, then MODFLOW-2000 will cease execution with an appropriate error message. In that case it may also be necessary to remove the *multiplier* array based on the FUNCTION. This is done, once again, by supplying the name of the array after a REMOVMULT keyword.

*Running FAC2MF2K*

As mentioned above, a settings file `settings.fig` must be present in the directory from which FAC2MF2K is run. Among other things, this file specifies the protocol to follow when reading and writing MODFLOW-compatible arrays from and to formatted files; in particular it specifies whether a "number of columns, number of rows" header is required in such a file.

Upon commencement of execution FAC2MF2K prompts:-

```
 Enter name of MODFLOW 2000 name file:-
```

Like MODFLOW itself, FAC2MF2K reads the names of all MODFLOW input files from this file. In particular, FAC2MF2K needs to know the names of the *sensitivity*, *parameter estimation*, *multiplier*, *zone*, *lpf* and/or *rch* files pertinent to the current MODFLOW run. Also, if an *asp* input file is cited in the name file (see below), it needs to know the name of this as well. Note that it is not essential that a *sensitivity* and *parameter estimation* file be amongst those comprising the input dataset for the current model. However if they are present, FAC2MF2K will read them (and modify the *sensitivity* file).

Next FAC2MF2K prompts:-

```
 Enter name of PPK2FAC-generated interpolation factor file:
 Is this a formatted or unformatted file? [f/u]:
```

This is the kriging factor file discussed above. As is indicated in the above prompt, it will have been prepared by running the PPK2FAC utility. Next FAC2FM2K asks:-

```
 Enter name of pilot points file [points_file]:
```

The default filename supplied with this prompt was read from the kriging factor file; this is the same pilot points file that was read by program PPK2FAC when it was run in order to calculate the kriging factors supplied in the file. You can use this same pilot points file when running FAC2MF2K (simply press the <Enter> key), or you can ask FAC2MF2K to read another pilot points file. If another file is read, the points cited in that file should be listed in the same order as in the pilot points file originally read by PPK2FAC. If not, FAC2MF2K will generate an appropriate error message.

The specifications of a pilot points file are set out in Section 2 of part A of this manual. FAC2FM2K obtains parameter values from the $5^{th}$ column of this file. These are the values that FAC2MF2K assigns to these parameters in the MODFLOW-2000 *lpf*, *rch* and *sensitivity* input files which it writes. If the *parameter estimation* process is active (or if the MODFLOW-2000 input dataset is converted to PEST format), then the hydraulic property values supplied in the pilot points file will be the initial values for use in the inversion process. (As stated above, upper and lower bounds used in the *parameter estimation* process are the same for all parameters of a particular type, these being provided by the variables MAXPARVAL and MINPARVAL in the parameter replacement file. You should ensure that the initial parameter values provided in the pilot points file are within these limits.)

FAC2MF2K next asks:-

```
 Enter name of integer array zonation file [integer_file]:
```

The default name supplied with the above prompt is the name of the zonation file read by PPK2FAC when it calculated kriging factors. Simply press the <Enter> key to accept the default (this being the appropriate course of action on most cases).

Next FAC2MF2K prompts for the name of the parameter replacement file. The specifications of this file are described in the preceding section. The prompt is:-

```
Enter name of parameter replacement file:-
```

FAC2MF2K's next prompt is:-

```
Enter name base for new multiplier and zonation arrays [3 chars or less]:
```

As has already been described, FAC2MF2K adds new *multiplier* arrays comprised of kriging factors to the MODFLOW-2000 *multiplier* file. It also adds the integer zonation array to the MODFLOW-2000 *zone* file. All new arrays added to the MODFLOW-2000 input dataset must have a name. The name given to the integer zonation array is the same as the 3-character string that you must supply in response to the above prompt. The names of the *multiplier* arrays are formed by adding a number to this 3-character string (if there is only one zone in the model domain, there can be as many of these arrays as there are pilot points). If there is any conflict between names formed in this fashion and existing *multiplier* and *zone* array names, FAC2MF2K will issue an appropriate error message informing you of this.

FAC2MF2K then prompts:-

```
Enter name for ASP input file:-
```

As is explained in Section 5 of Part A of this manual, the *asp* input file is read by the modified MODFLOW-2000 named MF2KASP. Note however that if MF2KASP is not supplied with an *asp* input file, or if the variables supplied in this file take on certain values, then operation of MF2KASP is identical to that of the normal MODFLOW-2000. MF2KASP is directed to its *asp* input file by an appropriate entry in the MODFLOW name file.

If an *asp* entry is already present in the name file read by FAC2MF2K, you must supply the name of this same name file to FAC2MF2K in response to the above prompt (FAC2MF2K will inform you if you don't). In this case FAC2MF2K gives you the option of overwriting the existing *asp* input file or of appending information to that file. In most cases the "overwrite" option should be selected; however see the section below entitled "Running FAC2MF2K Twice" for an exception to this recommendation. If the "overwrite" option is selected, FAC2MF2K supplies values for variables occurring in the first part of the *asp* input file which result in no modification to standard MODFLOW-2000 behaviour as far as PEST-interface, text output, and drying/re-wetting functionality is concerned ie. IPESTINT, LIMOP, HDRYBOT, NOSTOP and MINTHICK are all set to zero; these will be modified later to values more appropriate for use with PEST if you run the MODFLOW2000-to-PEST translator MF2PEST. However the INTERP variable occurring on the first data line of the *asp* input file will be set to 1, signifying that the *asp* input file contains information which modifies the operation of MODFLOW-2000 in building internal property arrays from parameter values; this information is written to the second part of the *asp* input file.

Modifications to MODFLOW's array-generation behaviour requested by the new *asp* input file are such that the instructions specified in the parameter replacement file and in the kriging factor file are obeyed. Thus if spatial parameter interpolation takes place on the basis of a log variogram, then log-based array generation is specified in

the *asp* input file for the pilot-point-based parameters added to the MODFLOW dataset by FAC2MF2K. Also, interpolation upper and lower bounds as set by the MININTERPVAL, MININTERPFILE, MAXINTERPVAL and MAXINTERPFILE variables supplied in the parameter replacement file are transferred to the *asp* input file.

When you have answered all of FAC2MF2K's prompts, it carries out its alterations to existing MODFLOW-2000 input files. FAC2MF2K undertakes the following tasks.

- Parameters identified for removal through the OUT keyword are removed from the *lpf*, *rch* and *sensitivity* process input files.

- Real and integer arrays identified for removal through the REMOVEMULT and REMOVEZONE keywords are removed from the MODFLOW-2000 *multiplier* and *zone* files.

- New pilot-point-based parameters are added to the *lpf*, *rch* and *sensitivity* files.

- New *multiplier* arrays containing kriging factors are added to the *multiplier* file.

- A new integer array containing pilot-point zonation is added to the *zone* file.

- If present, the *parameter estimation* input file is checked in order to ensure that no prior information is cited in that file pertaining to parameters which have been removed.

- An *asp* input file is written or modified.

- An *asp* entry citing the name of the *asp* input file is added to the existing MODFLOW name file.

However before any alterations are made to existing MODFLOW-2000 input files, FAC2MF2K copies all of these files to filenames of the same name, but with the extension ".old" attached. If, during any stage of FAC2MF2K execution, FAC2MF2K encounters an error condition then, before ceasing execution with an appropriate error message, it copies these stored files back to the original files so that the existing MODFLOW-2000 input dataset is unchanged. However if execution proceeds to completion without any problems, the names of all altered files, and the names of the files in which the original, unaltered dataset is stored, are written to the screen just before FAC2MF2K ceases execution.


*Running MODFLOW-2000*

The USGS version of MODFLOW will reject a name file which references an *asp* input file. So if a MODFLOW-2000 input dataset prepared by FAC2MF2K is to be used by the USGS version of MODFLOW, this entry must be removed from the namefile produced by FAC2MF2K. However you should carefully note the following.

- MODFLOW-2000 will not impose upper and lower limits in the array interpolation process; thus MININTERPVAL, MININTERPFILE, MAXINTERPVAL and MAXINTERPFILE settings in the parameter replacement file cannot be enforced if MODFLOW-2000 is used to carry out a simulation which includes pilot-point-based parameters.

- MODFLOW-2000 cannot carry out kriging based on a log variogram. If, when you ran PPK2FAC to generate kriging factors, you supplied a structure file containing a geostatistical structure in which the TRANSFORM value was "log", then kriging factors supplied in the kriging factor file apply to the logs of parameter values rather than to the parameter values themselves. These factors are transferred to MODFLOW-2000 as *multiplier* arrays. If the native MODFLOW-2000 is used, these factors will wrongly be applied to native parameter values instead of to log parameter values. Hence MF2KASP must be used if the factors contained in the *multiplier* arrays are to be properly applied. MF2KASP obtains information pertaining to the correct use of these factors from its *asp* input file - see Section 5 of Part A of this manual for further details.

- If array calculation is based on the logs of parameter values rather than on native parameter values, then sensitivities calculated by the MODFLOW-2000 *sensitivity* process will be in error. Thus although MF2KASP will happily calculated heads and all other MODFLOW output quantities, based on log-interpolation of parameter values, it cannot calculate sensitivities, and it cannot undertake parameter estimation.

In spite of the disadvantages that occur when interpolation is based on the logs of parameter values, the log-variogram option should still be selected when using pilot points to characterise a transmissivity or hydraulic conductivity distribution; many cases are documented in the literature where these hydraulic properties are best characterised by a log variogram.

The fact that MF2KASP cannot be used to undertake MODFLOW-2000 nonlinear parameter estimation for pilot-point-based parameters based on a log-variogram is really no disadvantage, for PEST can readily be used to undertake the parameter estimation process. A set of PEST input files can easily be prepared using the MODFLOW2000-to-PEST translator MF2PEST based on the MODFLOW-2000 input dataset written by FAC2MF2K. However if logarithmic interpolation is used, be sure to set the MF2PEST input variable IMDERCALC to 0 when running MF2PEST; this will instruct PEST to calculate MODFLOW derivatives using finite differences, rather than looking to MF2KASP to supply it with these derivatives. As was stated above, when log-based interpolation is undertaken, sensitivities (ie. derivatives) calculated by the MODFLOW-2000 *sensitivity* process are invalid. Hence there is no choice but to allow PEST to calculate these derivatives itself using finite parameter differences. (Don't forget to also set the HCLOSE solution convergence criterion to an appropriately low value as well; $10^{-4}$ is often quite appropriate.)

Perhaps the main advantage in using PEST to calibrate a model which is parameterised on the basis of pilot points is that PEST can be run in regularisation mode in order to apply geostatistically-based constraints on the allowed degree of spatial parameter variation. PEST's sophisticated regularisation functionality allows the user to set the degree to which PEST is prepared to sacrifice an optimal fit between model outputs and field data in order to accommodate these parameter variational constraints.. The use of pilot points, in conjunction with regularisation functionality, mostly allows better model-to-measurement fits to be obtained than can be obtained with discrete parameter zones, because a superfluity of such points can be used in order to allow PEST to introduce heterogeneity into the system where it is needed to obtain such a fit. The simultaneous imposition of regularisation constraints ensures that the calibrated property field is geologically reasonable. Furthermore, because of its beneficial effects in mitigating the deleterious effects of parameter insensitivity and correlation, the inversion process is numerically much more stable. Regularisation data can be added to a PEST control file written by MF2PEST using the PPKREG utility documented herein.

*Running FAC2MF2K Twice*

Sometimes it may be necessary to run FAC2MF2K more than once to complete the process of introducing pilot-point based parameters into a MODFLOW-2000 dataset. This will be necessary if parameters of different types are based on different sets of pilot points, or if interpolation takes place on the basis of a different variogram from the same set of pilot points for different parameter types. There is no problem in running FAC2MF2K more than once for this purpose. However be sure to provide FAC2MF2K with the name of the same *asp* input file on each occasion that you run it and, on each such occasion but the first, make sure to append data to this file rather than overwrite it.

**Uses of FAC2MF2K**

The usefulness of FAC2MF2K springs from the usefulness of characterising spatial variability of a hydraulic property through the use of pilot points supplemented by kriging as an interpolation mechanism. For a full discussion of this process, see Section 5 of Part A of this manual. As is explained in that section, two options are available to a modeller when implementing pilot-point-based parameterisation in the MODFLOW context. The first is to use FAC2REAL to build a MODFLOW-compatible real array on the basis of pilot points; MODFLOW must then be instructed to read the file containing this array as it commences execution. Parameters are defined as the values assigned to pilot points using a template file based on a pilot points file. The second mechanism is to use FAC2MF2K to modify an existing MODFLOW-2000 input dataset. That dataset can then be converted to PEST format using the MODFLOW2000-to-PEST translator, MF2PEST. In either case, geostatistically-based regularisation constraints can be introduced to the inversion process using the PPKREG utility.

**See Also**

See also FAC2REAL, PPK2FAC and PPKREG.

# FAC2REAL

**Function of FAC2REAL**

FAC2REAL undertakes the second stage of spatial interpolation to a model grid of the values assigned to a set of pilot points situated within (or close to) the model domain. The first stage of this spatial interpolation process is carried out by program PPK2FAC. PPK2FAC generates a set of cell-specific "kriging factors" by which hydraulic properties (or their logarithms) assigned to pilot points are multiplied prior to summation to form the interpolated value at each model cell. FAC2REAL carries out the actual multiplication and summation, writing the outcomes of its calculations to a MODFLOW-compatible real array file. Upper and lower limits can be applied to interpolated values if desired.

**Using FAC2REAL**

Execution of FAC2REAL is initiated by typing its name at the screen prompt. However it will cease execution immediately (with an appropriate error message) if a settings file (`settings.fig`) is not present within the directory from which it was run, and if this file does not possess a "`colrow`" descriptor, informing it whether or not formatted MODFLOW-compatible real array files begin with a "number of columns, number of rows" header line.

FAC2REAL's first prompts are:-

```
 Enter name of interpolation factor file:-
 Is this a formatted of unformatted file [f/u]:
```

The interpolation factor file will have been written by program PPK2FAC. It contains a set of kriging factors by which spatial interpolation is undertaken to cells within the finite difference grid. These kriging factors may be available for some, or all, of the cells of the model grid. Note that if an attempt is made to read an unformatted interpolation factor file as a formatted file (or vice versa), then FAC2REAL will cease execution with an error message.

FAC2REAL next prompts:-

```
 Enter name of pilot points file [points_file]:
```

A default filename is always supplied with this prompt. If the user responds to the prompt simply by pressing the <Enter> key, then the default filename is accepted. FAC2REAL obtains this filename from the interpolation factor file; when PPK2FAC writes this file, it records the name of the pilot points file from which it read pilot point coordinates and zone numbers.

The pilot points file read by FAC2REAL need not be the same pilot points file as that read by PPK2FAC when it was used to calculate kriging factors. *Nevertheless it must list the same points in the same order, and each point must be assigned to the same zone*. However the hydraulic property values assigned to the pilot points can be

different from those provided in the pilot points file read by PPK2FAC. (Note that if the number and/or ordering of points in the pilot points file supplied to FAC2REAL is different from that used in the pilot-points file supplied to PPK2FAC, FAC2REAL will detect the difference and cease execution with an appropriate error message.)

In carrying out spatial interpolation to the cell centres of a MODFLOW grid, FAC2REAL is able to impose an upper and lower limit on interpolated values. These limits can be globally applied, or they can be cell specific. FAC2REAL next prompts for information on these limits. First it asks:-

```
 Supply lower interpolation limit as an array or single value? [a/s]:
```

If you respond with an "s", FAC2REAL next prompts for the single lower interpolation limit applicable to the entire array.

```
  Enter lower interpolation limit:-
```

However if you respond to the above prompt with an "a", FAC2REAL prompts for the name of a file containing a MODFLOW-compatible real array.

```
  Enter name of lower interpolation limit array file:-
```

As usual, if the provided filename has an extension of "REF" FAC2REAL assumes that the file is formatted. However if it possesses an extension of "REU", FAC2REAL assumes that the file is unformatted. If the extension is neither of these, FAC2REAL prompts for the formatted/unformatted status of the file.

FAC2REAL then prompts for interpolation upper limits in a similar fashion.

Finally FAC2REAL asks for the name of the file to which it should write interpolated cell values. These are written in the form of a MODFLOW-compatible real array. FAC2REAL also asks for a value to provide to any cells in this array to which no interpolation takes place; this will occur if no factors pertaining to these cells are provided in the factor file - either because affected cells lie within zones for which interpolation factors were not calculated, or because they are too far away from any pilot points for kriging factors to have been evaluated. ("Too far" in this context is defined by the search radius supplied during execution of PPK2FAC.) An easily distinguished number such as $10^{35}$ often serves this purpose.

Once FAC2REAL has written its MODFLOW-compatible real array, it informs the user of this and ceases execution.

**Uses of FAC2REAL**

FAC2REAL runs quite quickly, for the spatial interpolation that it undertakes amounts to nothing more than the calculation of a few sums and products (with logarithmic transformation if kriging factors are based on geostatistical structures which pertain to the logs of hydraulic property values, rather than to the property values themselves). The bulk of the work required to carry out spatial interpolation

was actually undertaken by PPK2FAC; calculation of kriging factors can be a very time-consuming task if the number of pilot points is large.

Because FAC2REAL runs relatively quickly, there is little computational penalty involved in including it in a composite model, together with MODFLOW and/or MT3D, which is run by PEST as part of a parameter estimation process in which hydraulic property values are estimated at the locations of pilot points. Before running PEST, a template file must be created from a pilot points file. On every occasion that it runs the model during the parameter estimation process, PEST first writes a pilot points file based on the template file, using hydraulic property values that it wishes the model to use on that run. FAC2REAL then builds a MODFLOW-compatible real array using the values in the new pilot points file. This array can then be "pasted" into a MODFLOW input file using program REPARRAY; or it can be accessed by MODFLOW using its "OPEN/CLOSE" option for reading an external data file. After MODFLOW has run to completion, its outputs can be spatially and temporally interpolated to the sites and times of field measurements using utilities such as MOD2OBS and BUD2SMP (which should also be run by PEST as part of the "composite model"). Assistance in PEST input file preparation can be gained using program PESTPREP.

One advantage of undertaking pilot-points-based parameter estimation using this method instead of using MODFLOW-2000 for parameter definition and estimation, is that FAC2REAL can carry out spatial interpolation based on log-variograms, whereas MODFLOW-2000 cannot. Also, FAC2REAL can impose upper and lower limits on interpolated arrays (which can be very important on some occasions, especially if using a Gaussian variogram) where, once again, MODFLOW-2000 cannot. Note, however, that a specially modified version of MODFLOW (MODFLOW-ASP) does include such log-transformation and value-limiting functionality; see Section 5 of Part A of this manual for further details. However even if this version of MODFLOW-2000 is employed, the actual nonlinear parameter estimation process must still be undertaken by PEST for, under these circumstances, parameter sensitivities calculated by MODFLOW-2000 are invalid.

**See Also**

See FAC2MF2K, FIELDGEN, PPK2FAC and PPKREG.

# FEM2SMP

**Function of FEM2SMP**

FEM2SMP constitutes part of the PEST-MicroFEM interface. (Other programs in this interface are PPK2FACF and FAC2FEM.) FEM2SMP translates MicroFEM-generated heads and flows, contained within "fth" and "ftq" files, to bore sample file format; see part A of this manual for a description of this file format. Once model outputs are in this format, program SMP2SMP can be used to time-interpolate these outputs to the times at which measurements were made. (When calibrating a steady-state model, there will only be one such time.)

**Using FEM2SMP**

*General*

Like many other programs of the Groundwater Data Utilities, FEM2SMP requires that a "settings file" (named `settings.fig`) be present within the directory from which it is run. This file informs FEM2SMP whether the protocol for representation of dates is *dd/mm/yyyy* or *mm/dd/yyyy*; see part A of this manual for further details. Note also that you can "backtrack" in FEM2SMP execution by replying to any of its prompts with "E" or "e" (for "escape") followed by <Enter>. If FEM2SMP is being run as part of a "composite model" (encapsulated in a batch file) by PEST, then responses to its prompts can be placed in a text file; FEM2SMP is then directed to look to this file for the responses to its prompts by using the "<" character, followed by the name of the response file, as part of the FEM2SMP command issued from the batch file.

FEM2SMP commences execution with the prompts:-

```
Enter name of "fth" file (press <Enter> if none):
Enter name of "ftq" file (press <Enter> if none):
```

FEM2SMP will read either of both of these MicroFEM output files, transferring all of the information contained within them to the bore sample file which it generates. Next FEM2SMP prompts for the date and time at which the simulation begins:-

```
Enter simulation starting date [dd/mm/yyyy]:
Enter simulation starting time [hh:mm:ss]:
```

This information is required so that FEM2SMP can convert the elapsed simulation times recorded in the "fth" and "ftq" files into dates and times required for the bore sample file which it generates.

Finally FEM2SMP prompts for the name of the bore sample file which it must write:-

```
Enter name for new bore sample file:
```

Once it has been supplied with this information, FEM2SMP writes the bore sample file and terminates execution.

*Bore Identifiers*

As is explained in Part A of this manual, the first column of a bore sample file is comprised of bore identifiers (or the identifiers of other entities pertaining to sites at which measurements were made). These must be 10 characters or less in length. In converting MicroFEM outputs to bore sample file format, FEM2SMP must generate these names itself.

For a single layer model, it is an easy matter for FEM2SMP to generate bore (or site) identifiers. These are simply copied from the entity names provided in the "fth" and "ftq" files from which data is extracted; for "fth" files these entity names are equivalent to node labels. If any of these names are greater than 10 characters in length, FEM2SMP shortens them (from the left) to 10 characters. If the shortening of names in this manner results in nonuniqueness of bore identifiers (this contravening the rules governing construction of a bore sample file), FEM2SMP will warn the user of this before terminating execution. **Note that MicroFEM node labels should never contain spaces**.

The naming of bore identifiers becomes a more complicated task for multi-layer models. For such models MicroFEM allows node information from one or many layers to be stored in "fth" and "ftq" files. Where information from only one layer is recorded in these files (no matter how many layers are in the actual model), then the bore identifier naming convention employed by FEM2SMP is the same as has already been described. However where two or more layers are represented in these files, then unique identifier names are created through attaching a suffix to each MicroFEM entity name; this suffix is comprised of an underscore, followed by the name of the layer to which the information for that entity pertains. Thus, for example, the identifier "w13_3" refers to information from the node named "w13" recorded for layer 3. It should be noted that where suffixes are added to MicroFEM names to form bore identifiers in this manner, the chances of such an identifier exceeding the legal length of 10 characters is increased. As was mentioned above, shortening of names from the left is carried out in order that this protocol be respected; however incidences of name nonuniqueness are then more likely to arise.

*Matching of Names*

When MicroFEM is being calibrated by PEST, it is necessary to run SMP2SMP after FEM2SMP (which is run after MicroFEM) to undertake temporal interpolation of model outputs to the times at which measurements were made. SMP2SMP requires two bore sample files – an "observation sample file" and a "model sample file". The latter is produced by FEM2SMP while the former is supplied by the user. SMP2SMP matches bores (or sites) within these two files by their names. If this linkage is to be carried out correctly, then it is very important that bore (or site) identifiers in the observation bore sample file use the same naming convention as those in the model bore sample file. Thus in multi-layered modelling contexts where MicroFEM adds a layer number suffix to a user-supplied node name, this same suffix must be added to the bore name supplied in the observation bore sample file used by SMP2SMP.

**Uses of FEM2SMP**

Once model output data has been converted to bore sample file format by FEM2SMP, a number of postprocessing options for this data are available through various programs of the Groundwater Data Utilities. For example, SMP2HYD can be used to re-format this data for plotting using a graphing or spreadsheet package. However the most useful type of data processing in the calibration context is that provided by SMP2SMP which interpolates this data to the times at which measurements were made. Thus, after MicroFEM, FEM2SMP and SMP2SMP have been run, a model-generated equivalent to the user-supplied observation bore sample files exists. Respective entries in these files can be directly compared as they pertain to the same measurement times; differences between these entries can be minimised through the calibration process.

Spatial parameterisation of the MicroFEM model domain can be undertaken using pilot points. As many of these points as possible should be used, and the calibration process should be regularised. The "model input file", as far as PEST is concerned, then becomes one or more "pilot point files". A PEST template file must be created for each of these so that PEST can supply appropriate values for these points prior to each model run.

A batch file comprised of commands to run FAC2FEM, followed by MicroFEM followed by FEM2SMP, followed by SMP2SMP should be written. This then becomes the "model" to be run by PEST. Generation of a PEST instruction set to read the SMP2SMP output file, and of the PEST control file itself, can then be undertaken using PESTPREP. The PESTPREP-generated PEST control file may then require some slight user-modification; see the PESTPREP documentation in this manual for further details. It will also require the addition of regularisation prior information – a task carried out by PPKREG.

Note that if steady-state calibration is being undertaken, then only one item of information should be supplied for each bore or site in the observation bore sample file supplied to SMP2SMP. The "extrapolation threshold" supplied to SMP2SMP can then be supplied as an arbitrarily large number so that the single model output pertaining to each bore can be "interpolated" to the "measurement time", no matter what the time difference between the two.

**See Also**

See also FAC2FEM, PESTPREP, PPK2FACF, PPKREG, SMP2SMP.

# FIELDGEN

**Function of FIELDGEN**

FIELDGEN is a two-dimensional stochastic field generator in which field generation is undertaken using the Gaussian sequential simulation principal. FIELDGEN is more flexible than many other field generators. In particular:-

- FIELDGEN allows stochastic field generation to be zone-based; different fields can be generated in different zones within a model domain based on different geostatistical structures. The variograms comprising these geostatistical structures can be anisotropic if desired, with the axes of anisotropy oriented at arbitrary angles to the model grid.

- Fields generated by FIELDGEN can be conditioned by point measurements if desired.

- Fields can be generated for both uniform and nonuniform model grids.

Use of FIELDGEN allows a modeller to undertake stochastic MODFLOW/MODPATH/MT3D simulation. It also allows a modeller to undertake "calibration-constrained Monte-Carlo analysis" by combining stochastic field generation with pilot-point-based field multiplication under the control of PEST operating in regularisation mode.

**Using FIELDGEN**

*File Types Used by FIELDGEN*

FIELDGEN is a member of the pilot points family of programs. As such it requires file types which are typical of this suite, including a "pilot points file" and a "structure file". Both of these file types are discussed in Part A of this manual. When used with FIELDGEN the (optional) pilot points file contains conditioning data used in the stochastic field generation process. The structure file contains the variograms upon which stochastic field generation is based.

FIELDGEN obtains information on model domain zonation through a MODFLOW-compatible integer array. It writes the fields which it produces to a series of MODFLOW-compatible real array files.

*Sequential Gaussian Simulation*

The process of stochastic field generation by sequential simulation is very easy to understand. At each field point an expected field value and a field standard deviation pertaining to that point are first determined. These are calculated through kriging from points to which field values have already been assigned, as well as from points at which conditioning data exists (if available). Using the expected value and standard

deviation calculated in this way, a random field value is generated based on the assumption of a Gaussian probability distribution. The field value thus obtained can then be used in generating expected values and standard deviations at other field points at which field generation then takes place in the same way.

FIELDGEN provides the user with two kriging options, viz. simple and ordinary kriging. Simple kriging is preferred as this is more in harmony with the theoretical basis of sequential simulation. A number of other options are available for implementation of the kriging step. These are more fully discussed below; some of them can exert a significant influence on the computing time required for field generation.

The stochastic field generation engine of FIELDGEN is subroutine *sgsim* supplied with the GSLIB geostatistical software library; see Deutsch and Journel (1998) for further details.

*Model Grid and Stochastic Grid*

The GSLIB subroutine *sgsim* generates stochastic field values over a regular grid. Use of a regular grid facilitates optimisation of search strategies and other aspects of stochastic field generation. Groundwater model grids, however, are not always regular. A typical model grid may have a set of uniform, relatively small, cells disposed over that part of the model domain which is of particular interest; cell dimensions may then expand outwards from this area towards the model boundaries.

A problem which must be addressed in using *sgsim* for the generation of stochastic fields for a groundwater model is how to transfer field values calculated for the regular "stochastic grid" used by *sgsim* to the finite difference grid used by the model. If the model grid is uniform there is no problem, for FIELDGEN then ensures that the stochastic and model grids are coincident, thus allowing direct transfer of field values from the former to the latter grid. If the model grid is irregular, the transfer of field values is slightly more difficult.

As mentioned above, FIELDGEN allows the generation of stochastic field values independently within each zone of a model domain. Where cells within a particular zone are uniform, and where the dimensions of all cells within that zone are the same as those of the cell of minimum dimensions within the model domain, FIELDGEN will, as in the uniform grid case, ensure that the stochastic and model grids coincide. However if any cell within a particular zone has a row or column direction width that is greater than the minimum row or column direction width occurring anywhere within the model domain, then either of two different strategies can be used in transferring field values from the stochastic grid to the model grid. In the first of these strategies the field value assigned to a particular model cell is equated to that assigned to the stochastic grid node which is closest to its centre. In the second strategy the value assigned to a model cell is determined through averaging the values of all stochastic grid nodes lying within that cell (logarithmic average is undertaken for fields which are based on a log variogram). This strategy is useful where it is assumed that the variogram(s) used in field generation are based on a support area equal to that of the smallest cell in the model domain. Field values assigned to cells of greater area

are then automatically adjusted for the expansion of support area that occurs when working with these larger cells.

*Conditioning Data*

When undertaking stochastic field generation using the sequential simulation method it is an easy matter to incorporate independent measurements of the property being simulated into the field generation process. Such measurements are normally point-based; in the groundwater modelling context they will often consist of transmissivity or hydraulic conductivity values available from pump-test analyses. When calculating a local expected value and standard deviation at a stochastic grid node using kriging prior to random field generation at that node, these field measurements are simply included in the dataset from which kriging takes place.

If a conditioning point lies within a groundwater model cell, it does not follow that the enclosing cell will be assigned a value equal to the conditioning value. If the nugget pertaining to the geostatistical structure upon which stochastic field generation takes place is non-zero, then this will certainly not be the case. However if the nugget is zero and if the zone within which field generation is occurring contains uniform cells (with the cell dimensions being equal to those of the smallest cell in the model domain), and if a conditioning point falls exactly at the centre of a cell, then that cell will, in fact, be assigned a field value equal to the measurement value. But if, even under these uniform grid conditions where the stochastic and model grids coincide, a conditioning point does not lie exactly at the centre of a model cell, an expected value and standard deviation will be estimated at the cell centre through kriging based on conditioning points and field points for which values have already been assigned. Because the cell centre does not coincide with a conditioning point, the standard deviation at that cell centre will be nonzero. This will virtually ensure that the cell is not assigned a value equal to the measurement value at the nearby conditioning point. Nevertheless, the closer is the conditioning point to the cell centre, the closer will the generated value assigned to the cell be to the measured value associated with the conditioning point.

If the cell in which a conditioning point lies does not have dimensions equal to those of the smallest cell in the model domain, then a stochastic grid node will probably not lie at the model grid cell centre. Hence even if a conditioning point lies exactly at the model cell centre, then because that point does not exactly coincide with any particular stochastic grid node, the measurement value associated with the conditioning point can provide nothing more than a strong influence on field values generated for nearby stochastic grid nodes; it cannot determine the value assigned to any one of them exactly. When field values generated at stochastic grid nodes are then used to calculate model grid cell values by either of the two methods outlined above, a further diminution of the influence of the conditioning point on the field value assigned to a particular model cell can be suffered. Nevertheless, the conditioning point is still able to exert considerable influence on the value assigned to the model cell in which it lies; furthermore, any diminution of this influence takes place in a way that is in harmony with geostatistical principals.

*Variograms*

As is described in Section 5 of Part A of this manual and in the documentation of program PPK2FAC, each zone within a model domain can be characterised by a "geostatistical structure". This structure is comprised of an optional nugget, plus one or more variograms. The information required for this geostatistical characterisation is supplied in a "structure file". Four types of variogram can be featured in a structure file, viz. spherical, exponential, Gaussian and power.

The power variogram cannot be used in geostatistical field generation based on the sequential simulation principle due to the fact that it has no sill (and hence the field which it describes can potentially have infinite variance). Experience has demonstrated that problems can be encountered when using a structure which includes a Gaussian variogram; in particular, unusually high or low field values can be generated, and variance calculation appears to be a little unstable at times. Hence FIELDGEN allows the generation of stochastic fields using only two types of variogram, viz. the spherical and exponential variograms (ie. "type 1" and "type 2" variograms).

*Mean Field Values*

If no conditioning points are available for field generation within a particular zone, the user must supply FIELDGEN with the mean field value (ie. the expected field value) within that zone. Alternatively, if conditioning data is available, the mean can be determined from the conditioning data.

*Log Transformation*

It can be specified in the structure file that a geostatistical structure actually pertains to the log of a field, rather than to the field itself. However conditioning data is independent of the stochastic characterisation of a field. Thus conditioning measurements in the conditioning pilot points file must pertain to native field values irrespective of whether the geostatistical structure characterising that field pertains to the log of the field or to the native field. Likewise, MODFLOW-compatible real arrays generated by FIELDGEN will always contain native field values. However fields generated by FIELDGEN on the basis of log variograms are more informatively viewed after log-transformation of the respective arrays.

*Running FIELDGEN*

FIELDGEN is run by typing the command:

```
fieldgen
```

at the screen prompt. As for other members of the Groundwater Data Utilities, FIELDGEN will not run unless a settings file (named *settings.fig*) is present in the directory from which the above command is issued; see Section 1.5 of Part A of this manual.

FIELDGEN first prompts for the name of a grid specification file (see Section 2.9 of Part A of this manual). After it has read this file in order to determine the dimensions and geometry of the finite difference grid, it asks:-

```
Enter name of conditioning pilot points file (<Enter> if none):
```

The format of a pilot points file is discussed in Section 2.11 of Part A of this manual. The fourth column of this file contains integer values which define the model zone to which each pilot point pertains. The fifth column contains measurement values associated with pilot point locations, ie. the values used to condition field generation in the present case. If no conditioning data is to be used in stochastic field generation, respond to the above prompt by simply pressing the <Enter> key. (Note that if conditioning data is supplied, it is not necessary that it be supplied for all zones within a model domain; thus all zones do not need to be featured in the fourth column of the conditioning pilot points file.)

FIELDGEN's next prompt is:-

```
Enter name of zonal integer array file:
```

This is the file in which model domain zones are defined. It must contain a single integer array in which each zone is characterised by its own specific integer. After it has read the integer array, FIELDGEN prompts for the name of the structure file:

```
Enter name of structure file:
```

As mentioned above, the structure file contains definitions for one or a number of geostatistical structures characterising the variation of one or more hydraulic properties throughout one or a number of zones within the model domain. See Section 2.20 of Part A of this manual.

For each zone found in the integer zonation file, FIELDGEN issues the following prompts; possible responses are also shown below.

```
For zone characterised by integer value of n:-
Enter structure name (blank if no field generation for this zone): struct1
Use simple or ordinary kriging [s/o] in field generation: s
Enter maximum number of conditioning points to use: 40
Enter maximum number of previously simulated nodes to use: 40
```

As is apparent from the first of the above prompts, it is not necessary that stochastic field generation take place in every zone of the model domain (certainly not in inactive zones). However if stochastic field generation is desired for a particular zone, the name of a structure found in the structure file must be provided for that zone. As was mentioned above, the user is given the choice between simple or ordinary kriging for the calculation of expected values and standard deviations at stochastic grid nodes prior to field generation at these nodes. Kriging takes place on the basis of conditioning data (if it is provided for that zone) and field values already generated at other nodes. The user can select how many of each of these are used in the kriging process. If the computing time required to generate stochastic fields is inordinately large, these numbers should be reduced. Note that if no conditioning data is available for a zone, the user is not prompted for the number of conditioning points to use in the kriging process.

Next FIELDGEN asks:-

```
How many realizations do you wish to generate?
Enter filename base for real array files:
Write formatted or unformatted files?  [f/u]:
```

Supply an integer in response to the first of the above prompts. FIELDGEN will write each real array containing a stochastic field to a separate file. Each such file will be provided with a name comprised of the filename base (supplied by the user in response to the second of the above prompts), followed by the realisation number, followed by the extension ".*ref*" if the formatted file option is chose, or by an extension of ".reu" if the unformatted option is chosen.. As is explained in Section 2.17 of Part A of this manual, these are the default extensions used by members of the Groundwater Data Utilities for formatted and unformatted real array files respectively. Thus if the filename base supplied to FIELDGEN in response to the second of the above prompts is "*case*" and the formatted storage option is chosen, FIELDGEN will write a series of real array files named *case01.ref, case02.ref, case03.ref* etc.

If the model grid is non-uniform, FIELDGEN next asks:-

```
The model grid is non-uniform. To convert fields from stochastic
subgrid to model grid do you wish to average subgrid nodes or use
closest subgrid node?  [a/c]:
```

These options have been discussed above. Choice of the former (ie. "averaging") option is in accord with the affect of larger cell areas on the variogram, and the "smudging" of heterogeneity that the use of these larger cells entails. Use of the latter (ie. "closest") option will result in an overall variogram that more closely resembles the variogram used for field generation because no account is taken of the change in support caused by variable cell areas. It should be noted however that, as discussed above, if stochastic field generation is being undertaken in a zone comprised of uniform cells, and if the dimensions of these cells are those of the smallest cell in the model domain, then the stochastic subgrid will coincide exactly with the model subgrid in this zone; transferral of field values between the two grids is thus accomplished on a one-to-one basis.

Next, FIELDGEN prompts for the mean field value within each model domain zone in which stochastic field generation will take place. If conditioning data is available for a particular zone, it prompts:-

```
Enter mean field value in zone with integer value n
(Hit <Enter> to obtain this value from conditioning points in zone):
```

If you press <Enter> in response to this prompt, FIELDGEN will calculate the mean field value itself by averaging the values of conditioning data available for this zone; if a log variogram prevails in the zone, the logs of conditioning data are averaged rather than the native data. If no conditioning data is available for a particular zone, FIELDGEN does not write the second line of the above prompt; in this case the user has no option but to supply an average field value for the zone. Note that no such prompt is issued for zones in which no stochastic field generation takes place.

Finally FIELDGEN prompts:-

```
Enter integer seed for random number generator [324853]:
```

Press the <Enter> key to accept the default, or supply your own integer seed. For some applications the seed can be important. For example if you run FIEDGEN twice with exactly the same set of input data, including the seed, then the random fields which it generates will be identical on both runs. However if the seeds are different for the two different runs, then the arrays will be different.

At this stage FIELDGEN has received all of the information which it requires in order to undertake stochastic field generation. It then generates a stochastic field within each zone of the model domain for which such a field was requested; it does this *n* times, where *n* is the number of user-requested realisations, and writes the result to a real array file specific to each realisation. FIELDGEN records its activities to the screen as it works.

For complex model domains involving many zones the user is required to respond to many FIELDGEN prompts. Do not forget that, as is explained in Section 1.4 of Part A of this manual, you can retrace your steps (and correct any mistakes) at any stage simply through responding to any prompt with the single letter "e" (followed by <Enter>). FIELDGEN will then issue the previous prompt in its sequence of questions to the user. You can continue to retrace your steps in this fashion right back to the commencement of FIELDGEN execution.

**Uses of FIELDGEN**

Stochastic field generation is often used to explore the uncertainty in model predictions arising from hydraulic property heterogeneity. By undertaking repeated model runs based on different FIELDGEN-generated property fields, all of which respect any available conditioning data, and all of which respect what is known of the geostatistics of an area, the range of uncertainty associated with particular model predictions can be explored.

Where calibration constraints on parameter fields exist in the form of historical head and/or other measurements, the exploration of model predictive uncertainty using stochastic fields becomes more difficult. In this case a stochastic field must be "warped" before being used to make a prediction. "Warping" involves multiplication of a stochastic field by a maximally smooth "multiplier field", the latter being calculated through spatial interpolation between pilot points. Use of PEST in regularisation mode to determine this field (by estimating the values assigned to the pilot points upon which the field is based) ensures that the multiplier field deviates from homogeneity only to the minimum extent required to produce a "warped stochastic field" which calibrates the model. Thus the "warped" field retains as much of the stochastic structure of the original field as possible while respecting the constraints on this field imposed by the calibration process.

**See Also**

See also FAC2REAL and PPK2FAC.

**Reference**

Deutsch, C and Journel, A., 1998. GSLIB Geostatistical Software Library and User's Guide. Second Edition. Oxford University Press.

# GENREG

**Function of GENREG**

GENREG assists in the writing of a PEST input dataset where many parameters are estimated (often as a result of pilot point parameterisation of a model domain) through regularised inversion. It is assumed that a PEST control file already exists, and that this PEST control file cites all parameters that require estimation. It is also assumed that PESTMODE in this PEST control file is set to "regularisation" and that a "regularisation" section is present within the control file. However it is not necessary that any prior information be present within this file through which regularisation constraints are enforced. Nor is it necessary that any *regul\** observation groups be cited within this file. It is GENREG's task to add regularisation constraints to such a PEST control file and to assign them to regularisation groups.

It is assumed that parameters cited within the existing PEST control file belong to one or a number of different "families". In many instances different parameter families will pertain to different model layers, different hydrogeological units, and/or represent different hydraulic property types. Regularisation constraints can be formulated by GENREG on a family-by-family basis, between parameter families, or between members of a parameter family and a specific individual parameter. Furthermore, constraints can be of many different types. For example they can enforce intra-family homogeneity, or adherence to a single or spatially-variant preferred value. In all cases, a set of prior information equations representing these constraints is added to the "prior information" section of the existing PEST control file. (This section is created if it is not already present.) Weights are assigned to these new prior information equations according to a variety of different philosophies; for example weights can be uniform, geostatistically based, calculated as a function of proximity to the nearest observation point, or a combination of these.

While ostensibly a complex program, GENREG was written in order to simplify PEST setup for complex calibration problems involving regularised inversion. It achieves this by including in the one package the many different regularisation options offered through a variety of other utilities documented herein. Thus preparation for regularised inversion becomes a three-step process. In the first step the model domain is parameterised to a level of detail that is considered appropriate to the aims of the current modelling exercise. In the second step a PEST input dataset is constructed in which all estimable model parameters are cited. In the third step regularisation constraints are added to this PEST input dataset in order to render the inverse problem numerically tractable. These constraints may, or may not, take account of the pilot-point origin of many model parameters. Likewise they may, or may not, take account of the observation dataset provided for model calibration in the calculation of relative weighting between and within different regularisation groups. (It should not be forgotten that, when applied to regularisation groups, the *relative* rather than *absolute* weight is of most importance. PEST adjusts the contribution that prior information makes to the objective function itself as the regularisation process proceeds in accordance with the demands of that process.)

**Using GENREG**

*The Existing PEST Control File*

As mentioned above, use of GENREG assumes the existence of a PEST control file citing all parameters involved in the current parameter estimation problem. Prior to running GENREG, parameters which are to be fixed and those which are to be tied should be designated as such in the existing PEST control file. This is necessary because GENREG takes account of the tied/fixed status of model parameters when formulating prior information equations which embody the regularisation constraints which make solution of the highly parameterised inverse problem possible. In short, no constraint is formulated for a tied or fixed parameter; however the parent of a tied parameter, being estimable itself, should be subject to regularisation constraints.

The existing PEST control file should be as complete as possible. Optionally, it may include some prior information equations (which may, or may not, provide regularisation constraints and hence may, or may not, belong to observation groups whose names commence with the string *regul*). It should be internally consistent; thus the control variable NPAR should equal the number of cited parameters, NOBS should equal the number of cited observations, NOBSGP should equal the number of cited observation groups, NPRIOR should equal the number of prior information equations, etc. Furthermore, the PESTMODE variable should be set to "regularisation", and a "regularisation" section should exist within this file through which the values of regularisation control variables are supplied.

Notwithstanding its internal consistency, in most practical applications a pre-GENREG PEST control file will be declared as erroneous by PESTCHEK because of the absence of observation groups whose names begins with *regul* (and thus are used for the provision of regularisation constraints) in spite of the fact that PESTMODE is set to "regularisation". Hopefully, after GENREG has been run, this situation will have been rectified and PESTCHEK will give the resulting PEST control file clearance for takeoff.

*What GENREG Does*

GENREG modifies an existing PEST control file in some or all of the following ways.

- It adds a sequence of prior information equations pertaining to parameters cited within the "parameter data" section of the existing PEST control file. These equations take account of the transformation status of existing parameters; thus if a particular parameter is log-transformed, any new prior information equation that includes that parameter will reference the log of the parameter rather than the parameter itself.

- Weights are calculated for the newly-added prior information equations. These can be calculated in a variety of ways, some of which take into account the locations of adjustable parameters within the model domain (as well as the locations of observations on which their estimation is based).

- Each new equation is assigned to an observation group whose name is specified by the user; though GENREG does not insist on this, the names of such groups should mostly begin with the letters "regul". The observation groups to which new prior information equations are assigned may, or may not, already be cited in the existing PEST control file. If they are not, GENREG adds their names to the "observation groups" section of that file.

- "Problem size" variables such as NPRIOR and NOBSGP featured in the "control data" section of the existing PEST control file are amended as necessary in accordance with the updated contents of this file.

- If desired, the values assigned to regularisation control variables (found in the "regularisation" section of the PEST control file) are amended.

*Running GENREG*

GENREG has only three screen prompts. It first prompts for the name of an existing PEST control file. It then prompts for the name of a "GENREG control file". Finally it prompts for the name of the new PEST control file that it must write.

As is standard protocol for members of the PEST Groundwater Data Utility suite, if the user responds to any prompt by typing "e" followed by <Enter>, control is returned to the previous prompt. If "e" is supplied in response to the first prompt, GENREG terminates execution.

GENREG receives processing instructions from its control file, the details of which will be described below. Other files which GENREG must read may be cited within this file.

If GENREG encounters an error condition within any part of its input dataset, it will terminate execution with an appropriate error message. However it must be carefully noted that GENREG's checking of the existing PEST control file, and other aspects of its input dataset, is not nearly as thorough as is that of PESTCHEK. Thus it is possible that a GENREG-produced PEST control file may contain inherited inconsistencies; or it may contain inconsistencies which were introduced to this dataset though the provision of prior information to it. Hence a GENREG-produced PEST control file must always be checked with PESTCHEK before PEST is run on the basis of this file.

*The GENREG Control File*

An example of a GENREG control file is provided below.

```
###########################################
##      EXAMPLE OF A GENREG CONTROL FILE    #
###########################################


START REGULARISATION
  PHIMLIM 4.34
  PHIMACCEPT 4.40
  IREGADJ 1
END REGULARISATION


# Regularisation for layer 1 hydraulic conductivity.

START REGSPEC
  FAMILY_PREFIX           k_1
  SPEC_TYPE               within_family
  REG_TYPE                specified_value
  WEIGHT_TYPE             uniform
  WEIGHT_OBS_DIST         yes
  VAL_TYPE                file
  REG_GROUP               regul1
  WEIGHT_MULTIPLIER       3.0
  PILOT_POINTS_FILENAME   "k _1m.dat"
  OBS_COORD_FILENAME      "16 wells.crd"
  OBS_DIST_A              0.0
  OBS_DIST_b              1.0
  OBS_DIST_c              0.5
  OBS_DIST_MINWT          0.0
  OBS_DIST_MAXWT          1e10
END REGSPEC

# Regularisation for layer 1 specific yield.

START REGSPEC
  FAMILY_PREFIX           sf1_1
  SPEC_TYPE               within_family
  REG_TYPE                specified_value
  WEIGHT_TYPE             uniform
  VAL_TYPE                uniform
  VALUE                   0.1
  REG_GROUP               regul2
  WEIGHT                  1.0
END REGSPEC
```

**Part of a GENREG control file.**

When reading its control file, GENREG ignores blank lines and lines beginning with the "#" character.

The GENREG control file is subdivided into blocks. Each block must commence with the string "START *BLOCKNAME*" where "*BLOCKNAME*" must be either "REGULARISATION" (spelt with a "zee" instead of an "s" if the user insists) or "REGSPEC". It must end with the string END *BLOCKNAME*. These strings can be provided in upper or lower case.

Blocks can be supplied in any order. Only one REGULARISATION block can be provided within a GENREG control file; however there is no limit to the number of REGSPEC blocks which can be provided. Each such REGSPEC block contains the information which GENREG requires in order to add a set of prior information

equations to the existing PEST control file. Each GENREG control file must contain at least one REGSPEC block.

Within each block, data is provided using the "keyword" or "variable name" concept. Thus the name of a variable is supplied, followed by the value assigned to that variable. Depending on the variable, its value may be a real number, an integer or an ASCII string, including the name of a file. If a filename contains blanks, it must be surrounded by quotes.

Keywords within a block can be supplied in any order. GENREG reads all of these keywords before processing the block. It checks that all variables required for an identified processing task are present; if any are absent, GENREG reports this condition to the user before terminating execution. Keywords cited within a block that are not required for a particular processing task are simply ignored. Keywords and their values can be provided in upper or lower case.

In processing a block, GENREG undertakes the following tasks.

- After having read the entire block, GENREG checks that the information supplied in the block is complete.

- If required, GENREG then reads any files cited within the block; these may include pilot point files and observation coordinate files.

- If an error condition or inconsistency is detected either within the block or in any files cited therein, GENREG writes an appropriate error message to the screen, and then terminates execution.

- If the contents of the block are consistent and correct, GENREG generates a series of prior information equations, writing these to a temporary scratch file named *t###.###*.

- When GENREG has finished processing all of the blocks contained within its input dataset, the contents of this scratch file are transferred to the "prior information" section of the new PEST control file.

As will be discussed in detail below, the keywords that must be provided in a particular block depend on the type of processing that is instigated by that block. Hence some keywords must be provided in some blocks, but not in others. If any keywords that are necessary for implementation of the action specified in a particular block are absent from that block, GENREG will report this absence to the screen and then terminate execution. However if keywords are supplied which have no relevance to the type of processing undertaken by a particular block, these keywords are ignored by GENREG. Thus a user can easily make small modifications to the actions requested by the block without having to ensure that keywords made redundant by these modifications are removed from the block.

As is discussed in the PEST manual, each prior information equation appearing in the PEST control file must have a name. GENREG generates names itself for equations that it adds to the PEST input dataset. Each GENREG-generated prior information

equation is named *grn* where *gr* stands for "GENREG" and *n* is the number of the new prior information equation (with counting proceeding in the order in which these new equations are formulated by GENREG). In order to avoid name conflicts, the user should ensure that the names of any prior information equations already present within an existing PEST control file supplied to GENREG do not commence with *gr*.

### *The REGULARISATION Block*

The REGULARISATION block is optional; furthermore, only one such block can be present within a single GENREG control file. The purpose of this block is to allow alterations to be made to regularisation control variables supplied within the existing PEST control file. Values for regularisation variables supplied in the GENREG control file overwrite those supplied in the existing PEST control file.

Keywords that may appear in a REGULARISATION block, and the values that may be assigned to these keywords, are listed in the table below. Their names correspond to the names of regularisation control variables used by PEST; see the PEST manual for details.

| Variable name (i.e. keyword) | Allowed values | Suggested value |
| --- | --- | --- |
| PHIMLIM | a real number greater than zero | problem-specific |
| PHIMACCEPT | a real number greater than zero | normally about 1.05 times PHIMLIM |
| FRACPHIM | a real number less than unity (including zero, but not less) | 0.0 |
| MEMSAVE | "memsave" or "nomemsave" | "nomemsave" or omit |
| CONJGRAD | "cg" or "nocg" | "nocg" or omit |
| CGRTOL | a real number greater than zero and significantly less than unity | 1.0E-5; omit if CONJGRAD is omitted |
| CGITNLIM | an integer | 500; omit if CONJGRAD is omitted |
| WFINIT | a real number greater than zero | 1.0 |
| WFMIN | a real number less than WFINIT | 1.0E-10 |
| WFMAX | a real number greater than WFINIT | 1.0E10 |
| WFFAC | a real number greater than 1.0 | 1.3 |
| WFTOL | a real number greater than 0.0 | 1.0E-2 |
| LINREG | "linreg" or "nonlinreg" | "nonlinreg" |
| IREGADJ | 0, 1 or 2 | 1 |

**Keywords that can appear in a REGULARISATION block.**

Any of the keywords appearing in the above table can be omitted from the REGULARISATION block if desired. If the name of a variable is omitted, then the value of that variable supplied in the original PEST control file is transferred to the new PEST control file. In some instances, however, the inclusion of a variable in the REGULARISATION block of a GENREG control file may require that values be supplied for variables that do not already appear in the existing PEST control file. For example if CONJGRAD is set to "cg" in the GENREG control file (not presently recommended), but is absent from the existing PEST control file, values are also required for CGRTOL and CGITNLIM in the REGULARISATION block of the GENREG control file. If these are not supplied, GENREG will supply suitable default values.

*REGSPEC Block – General*

Each REGSPEC block presents the means by which a set of prior information equations is written providing regularisation constraints/linkages for a subset of the parameters appearing in the existing PEST control file. Parameters within the existing PEST control file are divided into "families"; each family is distinguished from other families by its "parameter prefix", which must be unique to that family. This prefix comprises the first *n* characters (where *n* is eight or less) of the names of all parameters belonging to that family.

Where parameters are linked to pilot points, and where processing undertaken by GENREG requires that a pilot points file pertaining to a family of parameters be read, the name of each parameter and the name of a pilot point to which it corresponds need not be the same. All that is required is that the pilot point name be the same as the parameter "root name" – that is, the name of the parameter minus its prefix. Alternatively, parameter names and pilot point names can correspond in full, for GENREG allows both of these alternatives.

Regularisation can be used to assign preferred values to parameters within an individual family, to formulate a pervasive set of equality relationships between members of the same family, to assign preferred values to differences or ratios of parameter values where the parameters involved in these relationships belong to different families, or to provide ratio or difference linkages between an individual parameter and all members of a parameter family. The nature of the relationships to be formulated by a particular REGSPEC block is supplied through the SPEC_TYPE and REG_TYPE keywords. Both of these keywords must be present in all REGSPEC blocks.

Three options are available for the SPEC_TYPE keyword, these being "within_family", "between_family" and "indiv_family". Each of these options is now discussed in detail.

*Within_Family Regularisation*

Two different types of regularisation can be applied to members of the same parameter family. The type of "within_family" regularisation undertaken by a particular REGSPEC block is set by the REG_TYPE keyword. Two values are

permitted for this keyword when SPEC_TYPE is set to "within_family", viz. "specified_value" and "equality".

REGSPEC block keywords (and their values) which can be used if SPEC_TYPE is set to "within_family" and REG_TYPE is set to "specified_value" are listed in the following table.

| Variable name (i.e. keyword) | Role | Possible values |
|---|---|---|
| SPEC_TYPE | sets regularisation specifications | "within_family" for the present table |
| REG_TYPE | sets regularisation type | "specified_value" for the present table |
| FAMILY_PREFIX | identifies subset of parameters for which regularisation is performed | a string of up to 8 characters |
| REG_GROUP | sets the regularisation group to which new prior information equations are assigned | a string of up to 12 characters (probably beginning with "regul") |
| WEIGHT_TYPE | determines how weights are calculated | must be "uniform" if REG_TYPE is "specified_value" |
| WEIGHT | weight assigned to all new prior information equations | a non-negative real number |
| WEIGHT_OBS_DIST | determines weather weights are multiplied by a factor that is dependent on distance to nearest observation point | "yes" or "no"; if this keyword is omitted it is assumed to be "no" |
| OBS_COORD_FILENAME | name of an observation coordinates file; required only if WEIGHT_OBS_DIST is set to "yes" | a text string, surrounded by quotes if it contains a space |
| WEIGHT_MULTIPLIER | the factor by which all weights for new prior information equations are multiplied; optional | a non-negative real number; assumed to be unity if omitted |
| OBS_DIST_A OBS_DIST_B OBS_DIST_C OBS_DIST_MINWT OBS_DIST_MAXWT | real numbers used in observation-distance weight factor calculation; required only if WEIGHT_OBS_DIST is set to "yes" | must be such that all calculated weights are non-negative |
| VAL_TYPE | determines how the "specified value" for each new prior information equation is obtained | "uniform" or "file" |
| VALUE | the specified value for all new prior information equations; required if VAL_TYPE is set to "uniform" | a real number |
| PILOT_POINTS_FILENAME | the name of a pilot points file; required if WEIGHT_TYPE is set to "obs_dist" and/or "value_type" is set to "file" | a text string, surrounded by quotes if it contains a space |

**Keywords required when SPEC_TYPE is "within_family" and "REG_TYPE" is "specified value".**

When SPEC_TYPE is set to "within_family", the parameter family for which regularisation constraints are formulated is identified by the FAMILY_PREFIX keyword. The regularisation group to which all new prior information equations will be assigned is designated through the REG_GROUP keyword. (Note that it makes sense to assign each new set of prior information equations to a new regularisation group; this allows PEST to conduct automatic inter-group regularisation weights adjustment in accordance with the setting of the IREGADJ regularisation control variable.)

Each "specified_value" prior information equation written by GENREG is of the form:-

```
pi_name 1.0 * param_name = specified_value obs_group weight
```

or

```
pi_name 1.0 * log(param_name) = log(specified_value) obs_group weight
```

The second of the above options is selected if the parameter cited in the prior information equation is log-transformed in the "parameter data" section of the existing PEST control file; the first is selected if the parameter is not log-transformed. No prior information equation is written for tied or fixed parameters.

The same preferred value can be assigned to all new prior information equations, or a preferred value can be assigned on a parameter-by-parameter basis. The first option is selected by setting VAL_TYPE to "uniform"; in that case a VALUE keyword must be supplied in order to designate the uniformly-applied specified value. Non-uniform specified values can be selected by setting VAL_TYPE to "file". In this case a PILOT_POINTS_FILENAME must be supplied, from which pilot point names, coordinates, zones and values are read. (Recall that a pilot points file must contain five columns of data; the first contains pilot point names, the second and third contain pilot point eastings and northings, the fourth contains pilot point zone numbers, while the fifth contains pilot point parameter values.) At least some (though not necessarily all) pilot point names within this pilot points file must correspond to parameters belonging to the currently-selected parameter family. As stated above, pilot point names are linked to parameter names either by equality of name, or by equality of the pilot point name to the parameter root name (i.e. the parameter name minus its family prefix); GENREG attempts to match parameter and pilot point names both ways. GENREG reads preferred parameter values from the fifth column of the pilot points file.

Each new prior information equation must be assigned a weight. GENREG requires that WEIGHT_TYPE be set to "uniform" when REG_TYPE is set to "specified_value"; the weight that is assigned to all new prior information equations is then supplied through the WEIGHT keyword. Optionally, the weight applied to each prior information equation can then be multiplied by a factor that is a function of the distance between the parameter cited in the prior information equation and the nearest observation point. (Increasing regularisation weights with distance from observation points can add stability to the regularised inversion process.) This option is selected by including a WEIGHT_OBS_DIST keyword in the REGSPEC block and setting its value to "yes". To de-activate this option, set WEIGHT_OBS_DIST to "no" or simply omit this keyword.

If WEIGHT_OBS_DIST is set to "yes", the name of both a pilot points file (from which parameter coordinates are read) and an observation coordinates file (from which observation coordinates are read) must be provided in the REGSPEC block. The first is provided through a PILOT_POINTS_FILENAME keyword, while the latter is provided through an OBS_COORD_FILENAME keyword. (Note that if VAL_TYPE is set to "file" GENREG will read preferred parameter values from this same pilot points file.)

An observation coordinates filename must have at least three columns of data. GENREG reads only the second and third columns of this file, assuming that observation point eastings lie within the second column and that observation point northings lie within the third column. Then, for each parameter belonging to the current family, it finds the distance between the corresponding pilot point and the closest observation location. It then calculates a weight factor for the pertinent specified-value prior information equation using the formula:-

$$weight\_factor = a + b*minimum\_distance**c$$

(where "**" stands for "raised to the power of"). *a*, *b* and *c* cited in the above equation are supplied through the OBS_DIST_A, OBS_DIST_B and OBS_DIST_C keywords respectively. Lower and upper limits can be imposed on weight factors calculated in this manner using the OBS_DIST_MINWT and OBS_DIST_MAXWT keywords; set these to zero and 1.0E20 respectively if you do not wish to impose such limits. All of these keywords must be supplied in a REGSPEC block if WEIGHT_OBS_DIST is set to "yes".

If desired, further weight multiplication can take place by assigning a value to the WEIGHT_MULTIPLIER keyword. Weights for all new prior information equations formulated by the REGSPEC block are multiplied by this multiplier, irrespective of whether distance-dependent multiplication has also taken place. If omitted, the WEIGHT_MULTIPLIER is assumed to be unity.

If REG_TYPE is set to "equality", then GENREG writes a series of prior information equations of the type:-

```
pi_name 1.0 * param1 - 1.0 * param2 = 0.0 obs_group weight
```

or

```
pi_name 1.0 * log(param1) - 1.0 * log(param2) = 0.0 obs_group weight
```

The latter type of equation is written if all adjustable (i.e. neither tied nor fixed) parameters in the selected family are log-transformed, while the former is written if all adjustable parameters are untransformed. Mixing of transformation types within the selected family is not allowed if REG_TYPE is set to "equality".

If "equality" regularisation is implemented, an EQUALITY_TYPE keyword must be present in the REGSPEC block. At present only two options are allowed for this keyword, viz. "next_pcf" and "spatial". In the former case parameters are linked through proximity in the "parameter data" section of the PEST control file. In the latter case, parameters are selected for equality constraint formulation based on spatial proximity.

The following table shows keyword options when EQUALITY_TYPE is designated as "next_pcf".

| Variable name (i.e. keyword) | Role | Possible values |
| --- | --- | --- |
| SPEC_TYPE | sets regularisation specifications | "within_family" for the present table |
| REG_TYPE | sets regularisation type | "equality" for the present table |
| EQUALITY_TYPE | sets method of selecting parameters for formulation of equality constraints | "next_pcf" for present table |
| FAMILY_PREFIX | identifies subset of parameters for which regularisation is performed | a string of up to 8 characters |
| REG_GROUP | sets the regularisation group to which new prior information equations are assigned | a string of up to 12 characters (probably beginning with "regul") |
| WEIGHT_TYPE | determines how weights are calculated | must be "uniform" if EQUALITY_TYPE is set to "next_pcf" |
| WEIGHT | weight assigned to all new prior information equations | a non-negative real number |
| WEIGHT_OBS_DIST | determines weather weights are multiplied by a factor that is dependent on distance to nearest observation point | must be set to "no", or omitted from REGSPEC block, if EQUALITY type is set to "next_pcf" |
| WEIGHT_MULTIPLIER | the factor by which all weights are multiplied for new prior information equations; optional | a non-negative real number; assumed to be unity if omitted |

**Keywords required when SPEC_TYPE is "within_family", "REG_TYPE" is "equality" and EQUALITY_TYPE is "next_pcf".**

If *n* parameters within the currently selected family are adjustable (i.e. neither tied nor fixed), then GENREG writes *n* prior information equations for that family when EQUALITY_TYPE is set to "next_pcf". Each such equation cites two parameters, these parameters being featured on subsequent lines of the "parameter data" section of the PEST control file. (If there are non-adjustable parameters, or parameters belonging to different families, mixed with parameters of the selected family, then such intervening lines are skipped when forging equality linkages in this manner.) Only uniform weighting is allowed; hence WEIGHT_TYPE must be set to "uniform" and a WEIGHT keyword must be supplied. A WEIGHT_MULTIPIER can be optionally supplied. However observation-distance-dependent weight factor calculation is not allowed; hence WEIGHT_OBS_DIST must be set to "no" or omitted.

More complex inter-parameter equality constraints can be introduced by GENREG if EQUALITY_TYPE is set to "spatial". In this case equality linkages, and the weights assigned to the prior information equations which enforce these linkages, are

determined by the spatial disposition of parameters. Because GENREG must know parameter locations, a PILOT_POINTS_FILENAME is an essential component of a REGSPEC block in which EQUALITY_TYPE is set to "spatial".

The following table lists REGSPEC keyword options when SPEC_TYPE is set to "within_family", REG_TYPE is set to "equality" and EQUALITY_TYPE is set to "spatial".

| Variable name (i.e. keyword) | Role | Possible values |
|---|---|---|
| SPEC_TYPE | sets regularisation specifications | "within_family" for the present table |
| REG_TYPE | sets regularisation type | "equality" for the present table |
| EQUALITY_TYPE | sets method of selecting parameters for imposing equality constraints | "spatial" for present table |
| FAMILY_PREFIX | identifies subset of parameters for which regularisation is performed | a string of up to 8 characters |
| REG_GROUP | sets the regularisation group to which new prior information equations are assigned | a string of up to 12 characters (probably beginning with "regul") |
| PILOT_POINTS_FILENAME | the name of a pilot points file | a text string, surrounded by quotes if it contains a space |
| WEIGHT_TYPE | determines how weights are calculated | "uniform", "sep_power", "sep_exp" or "sep_log" |
| WEIGHT | weight assigned to all new prior information equations; required only if WEIGHT_TYPE is set to "uniform" | a non-negative real number |
| SEARCH_RADIUS, MIN_PILOT_POINTS, MAX_PILOT_POINTS | determines number of equality linkages to which any individual parameter is subject | SEARCH_RADIUS is a positive real number: the other variables are integers |
| WEIGHT_SEP_A, WEIGHT_SEP_B, WEIGHT_SEP_C, WEIGHT_SEP_ANIS_BEARING, WEIGHT_SEP_ANIS_RATIO, WEIGHT_SEP_MAXWT, WEIGHT_SEP_MINWT | used in weight calculation when WEIGHT_TYPE is set to "sep_power", "sep_exp" or "sep_log" | real numbers |
| WEIGHT_OBS_DIST | determines weather weights are multiplied by a factor that is dependent on distance to nearest observation point | "yes" or "no"; assumed to be "no" if omitted |
| OBS_DIST_A    OBS_DIST_B OBS_DIST_C OBS_DIST_MINWT OBS_DIST_MAXWT | real numbers used in observation-distance weight factor calculation; required only if WEIGHT_OBS_DIST is set to "yes" | must be such that all calculated weights are non-negative |

| WEIGHT_MULTIPLIER | a factor by which all weights for new prior information equations are multiplied; optional | a non-negative real number; assumed to be unity if omitted |
|---|---|---|

**Keywords required when SPEC_TYPE is "within_family", "REG_TYPE" is "equality" and EQUALITY_TYPE is "spatial".**

Prior information equations generated by GENREG with EQUALITY_TYPE set to "spatial" are the same as those generated when EQUALITY_TYPE is set to "next_pcf". Similarly, it is a necessary prerequisite for formulation of equality constraints that all adjustable parameters belonging to a particular family be log-transformed or untransformed; mixed transformation types cannot be accommodated when forging parameter equality linkages.

Selection of parameters for inclusion in prior information equations which enforce equality constraints is based on the values assigned to the SEARCH_RADIUS, MAX_PILOT_POINT and MIN_PILOT_POINT keywords. For each parameter, a linkage is made to the closest MAX_PILOT_POINT parameters within a distance of SEARCH_RADIUS from that parameter. If less than MAX_PILOT_POINT parameters lie within this search radius, then only MAX_PILOT_POINT such linkages are forged. However if the number of linkages is less than MIN_PILOT_POINTS, GENREG ceases execution with an appropriate error message; in this case SEARCH_RADIUS will probably need to be increased (or MIN_PILOT_POINTS lowered – but it cannot be less than 1).

The weights assigned to new prior information equations can be uniform (in which case WEIGHT_TYPE should be set to "uniform" and a WEIGHT keyword should be supplied), or can be generated as a function of the separation between the two parameters cited within each equation. Three options are available for this latter method of weights calculation; one of them can be selected by setting the WEIGHT_TYPE keyword to "sep_power", "sep_exp" or "sep_log".

GENREG uses the following equation to generate weights using the "sep_power" option:-

$$weight = a + b * separation**c$$

where *separation* in the above equation is the distance between the two parameters featured in the prior information equation (possibly adjusted for anisotropy – see below). For the "sep_exp" option the equation is:-

$$weight = a + b * \exp(-c*separation)$$

while if WEIGHT_TYPE is assigned the string "sep_log", the equation is:-

$$weight = a + b * [\log10(separation)]**c$$

Values for *a*, *b* and *c* featured in the above equations must be assigned to the WEIGHT_SEP_A, WEIGHT_SEP_B and WEIGHT_SEP_C keywords. Note that GENREG insists that WEIGHT_SEP_C be positive if WEIGHT_TYPE is "sep_exp".

(Increasing weights with inter-parameter separation is not necessarily a bad idea, but should not be done exponentially.) GENREG will also object if two parameters occupy the same location and WEIGHT_TYPE is set to "sep_log".

As well as requiring values for WEIGHT_SEP_A, WEIGHT_SEP_B and WEIGHT_SEP_C, GENREG also requires values for WEIGHT_SEP_MAXWT, WEIGHT_SEP_MINWT, WEIGHT_SEP_ANISOTROPY_BEARING and WEIGHT_SEP_ANISOTROPY_RATIO if WEIGHT_TYPE is set to "sep_power", "sep_exp" or "sep_log". Weights calculated using any of the above three equations are limited from below using WEIGHT_SEP_MINWT and capped from above using WEIGHT_SEP_MAXWT. If it is desired that no such limitations be imposed on calculated weights, WEIGHT_SEP_MINWT should be set to 0.0 and WEIGHT_SEP_MAXWT should be set to a very high value (for example 1.0E20).

Sometimes it is desired that equality linkages be more strongly enforced in some directions than in others. This can be accomplished by providing GENREG with a WEIGHT_SEP_ANISOTROPY_RATIO which is greater than unity. (To avoid confusion, GENREG will not accept an anisotropy ratio value that is less than unity.). WEIGHT_SEP_ANISOTROPY_BEARING should be provided with a value that points in the direction of maximum parameter continuity, that is, in the direction in which equality linkages should be most strongly enforced. In this case GENREG multiplies distances in a direction perpendicular to this bearing by the value of WEIGHT_SEP_ANISOTROPY_RATIO. Thus "effective separations" are increased in the direction which is perpendicular to the axis of anisotropy. Hence if a weights calculation strategy is adopted that decreases weights with increasing inter-parameter separation, weights will decrease more rapidly in a direction perpendicular to the axis of anisotropy than along this axis.

Note that a WEIGHT_SEP_ANSISTROPY_BEARING of zero degrees is equivalent to north. Note also that setting WEIGHT_SEP_ANISOTROPY_RATIO to unity effectively disables anisotropy considerations; the value supplied for WEIGHT_SEP_ANISTROPY_BEARING then becomes redundant.

From the above discussion it is apparent that GENREG uses inter-parameter separations for two different purposes. One is for the calculation of weights using the above equations; the other is in the selection of points for which equality linkages are constructed, as determined by the SEARCH_RADIUS, MAX_PILOT_POINT and MIN_PILOT_POINT keywords. Modification of inter-parameter distances through the adoption of a non-unity anisotropy ratio affects both of these processes. **Thus the search radius is shortened in the direction perpendicular to the axis of anisotropy** as supplied through the WEIGHT_SEP_ANISOTROPY_BEARING keyword.

Whether weights for new prior information equations are assigned a uniform value, or are calculated on the basis of inter-parameter separation, these weights can optionally be multiplied by a factor that depends on the distance between parameters involved in each such linkage and the nearest observation point. For each of the two parameters cited in any new prior information equation, GENREG calculates the distance to the nearest observation point; observation point coordinates are supplied through an

OBS_COORD_FILENAME file. The higher of these two distances is used in weight factor calculation on the basis of the OBS_DIST_A, OBS_DIST_B, OBS_DIST_C, OBS_DIST_MINWT and OBS_DIST_MAXWT keywords in the manner already described. Note that the latter two variables limit the weight *factor*, rather than the calculated weights themselves. Note also that such observation-distance-based modification of weights will only occur if the optional WEIGHT_OBS_DIST keyword is set to "yes". It should be further noted that observation-parameter distances are not affected by a non-unity anisotropy ratio.

If desired, weights can undergo final modification using the optional WEIGHT_MULTIPLIER keyword. This is applied uniformly to all new prior information equations generated by the REGSPEC block, irrespective of how these weights were calculated.

*Between_Family Regularisation*

GENREG allows prior information equations to be written expressing relationships between members of different parameter families. As for "within-family" regularisation, a number of different options exist for the formulation of "between-family" regularisation constraints. However, in any one GENREG block, only two parameter families can be selected for the formulation of these constraints. These are identified through the PARAMETER_PREFIX_1 and PARAMETER_PREFIX_2 keywords.

A pre-requisite to the generation of a set of "between-family" prior information equations is that all adjustable (i.e. untied and unfixed) members of the selected parameter families be either log-transformed or untransformed. The user must provide a ratio or difference which is then applied to pairs of parameters (one member of each pair is taken from each family) through a prior information equation of the type:-

```
pi_name 1.0 * par1 - 1.0 * par2 = difference obs_group weight
```

or

```
pi_name 1.0 * log(par1) - 1.0 * log(par2) = log(ratio) obs_group weight
```

As is apparent from the above equations, a parameter difference is applicable when members of both selected parameter families are untransformed, whereas a parameter ratio is appropriate where the parameters belonging to both families are log-transformed; in the latter case a difference equation is written with respect to the logs of the pertinent parameters, this difference being equated to the log (to base 10) of the user-supplied parameter ratio. This ratio must be greater than zero or GENREG will object.

For the purpose of generating prior information equations, parameters from different families can be matched either by name or by spatial proximity. This selection is made through the DIFFRAT_TYPE keyword. ("DIFFRAT" is short for "DIFFerence or RATio".) The table below shows permissible keywords and keyword values when the "name" option is selected.

| Variable name (i.e. keyword) | Role | Possible values |
|---|---|---|
| SPEC_TYPE | sets regularisation specifications | "between_family" for the present table |
| REG_TYPE | sets regularisation type | "difference" or "ratio" for the present table |
| DIFFRAT_TYPE | sets parameter selection method for formulation of difference or ratio prior information equations | "name" for the present table |
| DIFFRAT_VAL_TYPE | sets source of difference or ratio | "uniform" or "file" |
| VALUE | sets difference or ratio if DIFFRAT_VAL_TYPE is set to "uniform" | a real number; this number must be greater than zero if REG_TYPE is set to "ratio" |
| PILOT_POINTS_FILENAME_DR | the name of a pilot points file from which differences or ratios are read if DIFFRAT_VAL_TYPE is set to "file"; also used for parameter locations if WEIGHT_OBS_DIST is set to "yes" | fifth column of this table must contain real numbers; these numbers must be greater than zero if REG_TYPE is set to "ratio"; second and third columns must contain pilot point eastings and northings. |
| FAMILY_PREFIX_1 | identifies subset of parameters for inclusion in first parameter family | a string of up to 8 characters |
| FAMILY_PREFIX_2 | identifies subset of parameters for inclusion in second parameter family | a string of up to 8 characters |
| REG_GROUP | sets the regularisation group to which new prior information equations are assigned | a string of up to 12 characters (probably beginning with "regul") |
| WEIGHT_TYPE | determines how weights are calculated | must be "uniform" for "between_family" regularisation |
| WEIGHT | weight assigned to all new prior information equations | a non-negative real number |
| WEIGHT_OBS_DIST | determines whether weights are multiplied by a factor that is dependent on distance to nearest observation point | "yes" or "no"; assumed to be "no" if omitted |
| OBS_DIST_A    OBS_DIST_B OBS_DIST_C OBS_DIST_MINWT OBS_DIST_MAXWT | used in observation-distance weight factor calculation; required only if WEIGHT_OBS_DIST is set to "yes" | real numbers; must be such that all calculated weights are non-negative |
| WEIGHT_MULTIPLIER | the factor by which all weights for new prior information equations are multiplied; optional | a non-negative real number; assumed to be unity if omitted |

**Keywords required when SPEC_TYPE is "between_family", "REG_TYPE" is "difference" or "ratio", and DIFFRAT_TYPE is "name".**

If DIFFRAT is set to "name", then the parameter families selected through the FAMILY_PREFIX_1 and FAMILY_PREFIX_2 keywords must meet certain criteria. These are:-

- each family must possess the same number of adjustable parameters;

- the names of these parameters must be identical between families except for the respective family prefix;

- all adjustable members of both families must be either untransformed or log-transformed.

If there are *n* adjustable parameters in either selected parameter family, GENREG writes *n* prior information equations of the type illustrated above. Parameters appearing in each equation are linked by name; that is, except for the family prefix, the names of the two parameters appearing in any one prior information equation will be identical. It is important to note that in each of these equations the parameter selected through FAMILY_PREFIX_1 appears first, while the parameter selected through FAMILY_PREFIX_2 appears second. The difference or ratio always applies to the parameters in this order.

The value of the difference or ratio can be the same for all parameter pairs, or can be assigned on a parameter-by-parameter basis. In the first case DIFFRAT_VAL_TYPE should be set to "uniform", and the difference or value supplied through the VALUE keyword. In the second case DIFFRAT_VAL_TYPE should be set to "file" and the name of a pilot points file should be assigned to the PILOT_POINTS_FILENAME_DR keyword. Pilot point names within this file should be the same as the names of parameters belonging to the first parameter family. Alternatively, pilot point names should be the same as parameter root names (i.e. the parameter name without the family prefix); in the latter case pilot point names are therefore similarly linked to the members of the second parameter family.

When "between_family" regularisation is implemented using the "name" option, WEIGHT_TYPE must be set to "uniform"; the weight to be used in "between-family" prior information equations is then supplied through the WEIGHT keyword. WEIGHT_OBS_DIST can optionally be set to "yes", in which case prior information weights supplied in this manner are multiplied by a factor that is a function of the distance between parameters featured in a given prior information equation and the nearest observation point. When DIFFRAT_TYPE is set to "name", it is assumed that both parameters featured in any new prior information equation have the same easting and northing, these being read from the PILOT_POINTS_FILENAME_DR file. As stated above, points within this file can be linked by name to parameters of the first parameter family, or by parameter root name to both parameter families. Note that where WEIGHT_OBS_DIST is set to "yes" and DIFFRAT_VAL_TYPE is set to "file", the PILOT_POINTS_FILENAME_DR file is used twice by GENREG – once to obtain the values of parameter-by-parameter differences or ratios, and once to obtain parameter coordinates for use in WEIGHT_OBS_DIST prior information weight modification.

A second major processing option for "between_family" regularisation can be implemented by setting DIFFRAT_TYPE to "spatial". In this case GENREG does not match parameters from the two selected parameter families by name; rather it matches them by spatial proximity. Hence a pilot points file must be read by GENREG for each parameter family. The names of these files are supplied through the PILOT_POINTS_FILENAME_1 and PILOT_POINTS_FILENAME_2 keywords; parameter families associated with these pilot point files must correspond to the FAMILY_PREFIX_1 and FAMILY_PREFIX_2 families respectively. (As usual, when reading a pilot points file, GENREG attempts to match pilot points to parameters using full parameter and pilot point names and, if this is not successful, pilot point names and parameter root names.)

For the "spatial" DIFFRAT_TYPE option, DIFFRAT_VAL_TYPE must be set to "uniform". Thus all prior information equations formulated by the GENREG block will feature the same number on the right hand side of the equality sign. As for the "name" option, all adjustable parameters in both families must be either log-transformed or not transformed at all. In the former case REG_TYPE must be supplied as "ratio" and in the latter case it must be supplied as "difference".

Keyword options for the "spatial" DIFFRAT_TYPE alternative are set out in the table below.

| Variable name (i.e. keyword) | Role | Possible values |
|---|---|---|
| SPEC_TYPE | sets regularisation specifications | "between_family" for the present table |
| REG_TYPE | sets regularisation type | "difference" or "ratio" for the present table |
| DIFFRAT_TYPE | sets parameter selection method for formulation of difference or ratio prior information equations | "spatial" for the present table |
| DIFFRAT_VAL_TYPE | sets source of difference or ratio | must be "uniform" if DIFFRAT_TYPE is set to "spatial" |
| VALUE | sets difference or ratio | a real number; this number must be greater than zero if REG_TYPE is set to "ratio" |
| FAMILY_PREFIX_1 | identifies subset of parameters for inclusion in first parameter family | a string of up to 8 characters |
| FAMILY_PREFIX_2 | identifies subset of parameters for inclusion in second parameter family | a string of up to 8 characters |
| PILOT_POINTS_FILENAME_1 | the name of a pilot points file from which eastings and northings for first parameter family are read | a filename text string |

| PILOT_POINTS_FILENAME_2 | the name of a pilot points file from which eastings and northings for second parameter family are read | a filename text string |
|---|---|---|
| SEARCH_RADIUS | distance from a pilot point in one family for which a match is sought from other family | a real number greater than zero |
| MAX_PILOT_POINTS | maximum number of inter-family linkages to create from a member of one family to members of the other family | an integer greater than zero |
| MIN_PILOT_POINTS | GENREG will issue either an error or warning message if this number of parameters from one parameter family is not found within one search radius of a member of the other family | an integer greater than zero |
| WARN_LESS_MIN | issue a warning rather than error message if MIN_PILOT_POINTS members of one family are not found within one SEARCH_RADIUS of a member of the other family | "yes" or "no"; if omitted "no" is assumed and an error message is issued prior to cessation of GENREG execution |
| REG_GROUP | sets the regularisation group to which new prior information equations are assigned | a string of up to 12 characters (probably beginning with "regul") |
| WEIGHT_TYPE | determines how weights are calculated | must be "uniform" for "between_family" regularisation |
| WEIGHT | weight assigned to all new prior information equations | a non-negative real number |
| WEIGHT_OBS_DIST | determines weather weights are multiplied by a factor that is dependent on distance to nearest observation point | "yes" or "no"; assumed to be "no" if omitted |
| OBS_DIST_A OBS_DIST_B OBS_DIST_C OBS_DIST_MINWT OBS_DIST_MAXWT | real numbers used in observation-distance weight factor calculation; required only if WEIGHT_OBS_DIST is set to "yes" | must be such that all calculated weights are non-negative |
| WEIGHT_MULTIPLIER | the factor by which all weights are multiplied for new prior information equations; optional | a non-negative real number; assumed to be unity if omitted |

**Keywords required when SPEC_TYPE is "between_family", "REG_TYPE" is "difference" or "ratio" and DIFFRAT_TYPE is "spatial".**

In forming parameter linkages for the purpose of writing prior information equations, GENREG first traverses the elements of the first parameter family. For each member of this family it finds all parameters from the second family that lie within one SEARCH_RADIUS of this parameter. It then selects the closest

MAX_PILOT_POINTS of these second-family parameters for the writing of prior information equations. If fewer than this number of pilot points lie within one search radius of the first-family parameter, it simply uses the parameters that it finds. However if less than MIN_PILOT_POINTS second-family parameters are found within this radius it either warns the user of this, or ceases execution with an appropriate error message. The user selects between these modes of GENREG behaviour through the WARN_LESS_MIN keyword. If this keyword is omitted or set to "no", the error message alternative is chosen.

After having traversed the list of parameters comprising the first parameter family, GENREG traverses the list of parameters belonging to the second parameter family. The parameter search process is repeated. However in generating prior information equations on the basis of this second sweep, GENREG does not re-write equations that arose out of its first parameter sweep; it only generates a new equation if traversal of the second parameter list results in a new between_family parameter linkage being formed.

As for the "name" DIFFRAT_TYPE option, WEIGHT_TYPE must be set to "uniform", and the weight itself supplied through the WEIGHT keyword. However weight multiplication based on distance to observation points will be undertaken if WEIGHT_OBS_DIST is set to "yes". In this case, values must be assigned to all of the OBS_DIST_A, OBS_DIST_B, OBS_DIST_C, OBS_DIST_MINWT and OBS_DIST_MAXWT keywords. Alternatively, if WEIGHT_OBS_DIST is set to "no" or omitted, these latter keywords can also be omitted. Note that when evaluating the parameter-to-observation point distance on which this weight factor calculation is based, GENREG first calculates the distance to the nearest observation point from both parameters cited in each new prior information equation. It then uses the higher of these two distances as a basis for weight factor calculation. Note also that, as discussed above, the OBS_DIST_MINWT and OBS_DIST_MAXWT keywords are used to constrain the weight *factor* calculated through the WEIGHT_OBS_DIST equation, rather than the weight itself.

As usual, a WEIGHT_MULTIPLER keyword can be used for final multiplication of prior information weights after all other weight processing has taken place.

The user may be wondering why the WARN_LESS_MIN option is available for "between_family" prior information equation generation under the "spatial" option, but is not available for "within_family" spatially-based prior information equation generation. In most cases where "within_family" regularisation is undertaken, it is applied to parameters comprising a single horizontal or sub-horizontal model layer or hydrgeologic unit. However "between_family" regularisation is normally invoked in order to stabilize the estimation of parameters in separate layers, with each family being assigned to a separate layer or unit. In the former case, it is important that all members of the parameter family be included in at least one prior information equation. In the latter case, members of each family will probably already be involved in at least one "within_family" prior information equation by virtue of their intra-layer or intra-unit status. Hence the need for "vertical regularisation" at spatial locations where no layer overlap occurs may be diminished.

## *Indiv_Family Regularisation*

GENREG allows the generation of a set of prior information equations expressing a difference or ratio relationship between all members of a particular parameter family and a single adjustable parameter which is not a member of any family, but which is nevertheless featured in the PEST control file. To activate this mode of prior information equation generation SPEC_TYPE must be set to "indiv_family". Other keywords pertinent to "indiv_family" regularisation, and their allowed assignments, are depicted in the table below.

| Variable name (i.e. keyword) | Role | Possible values |
|---|---|---|
| SPEC_TYPE | sets regularisation specifications | "indiv_family" for the present table |
| REG_TYPE | sets regularisation type | "difference" or "ratio" for the present table |
| DIFFRAT_VAL_TYPE | sets source of difference or ratio | "uniform" or "file" |
| VALUE | sets difference or ratio if DIFFRAT_VAL_TYPE is set to "uniform" | a real number; this number must be greater than zero if REG_TYPE is set to "ratio" |
| FAMILY_PREFIX | identifies subset of parameters for inclusion in current parameter family | a string of up to 8 characters |
| PARAMETER | identifies individual parameter to which all adjustable parameters in the identified family are linked by a prior information equation | a string of up to 12 characters |
| PILOT_POINTS_FILENAME_DR | the name of a pilot points file from which differences or ratios are read if DIFFRAT_VAL_TYPE is set to "file"; also used for parameter locations if WEIGHT_OBS_DIST is set to "yes" | fifth column of this table must contain real numbers; these numbers must be greater than zero if REG_TYPE is set to "ratio"; second and third columns must contain pilot point eastings and northings |
| REG_GROUP | sets the regularisation group to which new prior information equations are assigned | a string of up to 12 characters (probably beginning with "regul") |
| WEIGHT_TYPE | determines how weights are calculated | must be "uniform" for "indiv_family" regularisation |
| WEIGHT | weight assigned to all new prior information equations | a non-negative real number |
| WEIGHT_OBS_DIST | determines weather weights are multiplied by a factor that is dependent on distance to nearest observation point | "yes" or "no"; assumed to be "no" if omitted |
| OBS_DIST_A  OBS_DIST_B OBS_DIST_C OBS_DIST_MINWT OBS_DIST_MAXWT | real numbers used in observation-distance weight factor calculation; required only if WEIGHT_OBS_DIST is set to "yes" | must be such that all calculated weights are non-negative |

| WEIGHT_MULTIPLIER | the factor by which all weights are multiplied for new prior information equations; optional | a non-negative real number; assumed to be unity if omitted |
|---|---|---|

**Keywords required when SPEC_TYPE is "indiv_family".**

If SPEC_TYPE is set to "indiv_family", REG_TYPE must be set to "difference" or "ratio". In the former case both the identified individual PARAMETER and all members of the currently-identified parameter family that are neither tied nor fixed must be untransformed in the "parameter data" section of the PEST control file. In this case, GENREG generates a prior information equation of the following type for each adjustable member of the current parameter family:-

```
pi_name 1.0 * par1 – 1.0 * parameter = difference obs_group weight
```

Note that the member of the current parameter family is featured first in this equation and the individual parameter to which it is linked is featured second; the difference thus pertains to these two parameters in that order.

Where REG_TYPE is set to "ratio", GENREG insists that the individual PARAMETER be log transformed, and that all non-fixed and non-tied members of the identified parameter family also be log transformed in the PEST control file. In this case GENREG generates prior information equations of the type:-

```
pi_name 1.0 * log(par1) – 1.0 * log(parameter) = log(ratio) obs_group weight
```

If DIFFRAT_VAL_TYPE is set to "uniform" then the same difference or ratio is used in each new prior information equation, this being supplied through the VALUE keyword. Alternatively if DIFFRAT_VAL_TYPE is set to "file", the parameter-specific difference or ratio is read from the fifth column of the pilot points file whose name is assigned to the PILOT_POINTS_FILENAME_DR keyword. In this file, pilot point names are linked to parameter names either by full name, or by parameter root name; that is, a pilot-point-to-parameter match is made if the name of a pilot point is the same as the name of a parameter, or has the same name as a parameter with the latter's FAMILY_PREFIX removed.

Whether or not DIFFRAT_VAL_TYPE is set to "file", a PILOT_POINTS_FILENAME_DR keyword is required if WEIGHT_OBS_DIST is set to "yes". In this case weights are multiplied by a factor which is calculated as a function of the separation between a particular parameter and the closest observation point. Parameter coordinates are obtained from the second and third columns of the PILOT_POINTS_FILENAME_DR file, while observation coordinates are read from the second and third columns of an observation coordinates file whose name is assigned to the OBS_COORD_FILENAME keyword. In this case values must also be assigned to the OBS_DIST_A, OBS_DIST_B, OBS_DIST_C, OBS_DIST_MINWT and OBS_DIST_MAXWT keywords in the current REGSPEC block. Irrespective of the WEIGHT_OBS_DIST setting, WEIGHT_TYPE must be set to "uniform" and a non-negative weight must be assigned to the WEIGHT keyword (WEIGHT_OBS_DIST functionality does not replace the uniform WEIGHT assigned in this manner; instead, it multiplies it by a distance-dependent factor.) The optional WEIGHT_MULTIPLIER keyword can be used to supply yet another weight multiplication factor.

As for other REGSPEC blocks, the REG_GROUP keyword must be supplied with the name of the observation group to which new prior information equations are to be assigned.

**Uses of GENREG**

GENREG is used to assist in the construction of a complex PEST input dataset implementing regularised inversion. Because it presents so many options for the inclusion of regularisation constraints within a PEST control file, and because these options can be invoked (and varied) so easily, it allows the user to experiment with different regularisation schemes in order to find one that is most suitable for his/her particular model calibration problem.

GENREG is best used in combination with PARM3D. Using the latter program a complex model domain comprised of many different model layers. and possibly many different hydrostratigraphic units, can be parameterised using pilot points in a straightforward manner.

The use of many different parameters in the inversion process allows maximum information content to be extracted from a given calibration dataset. However this cannot be achieved without introducing some kind of regularisation device to the inverse problem. One such device is comprised of a set of constraints on parameter values which define a "preferred system condition" from which deviation will be tolerated only to the extent necessary to achieve a user-specified level of model-to-observation fit. Another device is the use or singular value decomposition to limit, in a more direct way, the number of degrees of freedom that can be represented in a calibrated parameter field. And, of course, these two methodologies can be combined through PEST's unique and powerful SVD-assist functionality.

GENREG was written to complement PEST's high-end regularisation functionality by providing a modeller with the wherewithal to access that functionality as easily as possible. As such, it is envisaged that it will find widespread use, and that as a result of experience gained through such use, its capabilities will be expanded over time.

**See Also**

See also PPKFAC, FAC2REAL and PARM3D.

# GRID2ARC

**Function of GRID2ARC**

Program GRID2ARC produces a pair of ARCINFO "generate" files based on a user-defined portion of the finite difference grid. These files can be used to reconstruct the model grid within ARCINFO, with a polygon defined for each cell.

**Using GRID2ARC**

A settings file `settings.fig` must be present in the directory from which GRID2ARC is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

GRID2ARC commences execution with the prompt:

```
 Enter name of grid specification file:
```

Type in the name of the grid specification file. Alternatively, if a filename file (`files.fig`) is present in your current directory, GRID2ARC may present you with a default grid specification filename (as read from the filename file); in this case press <Enter> to accept the default or type in the correct name. GRID2ARC reads the grid specification file in order to obtain the geographical information which it requires for construction of the ARCINFO "generate" files.

Next GRID2ARC prompts:

```
 Enter name of window integer array file:
```

to which you should respond with the name of an appropriate MODFLOW/MT3D-compatible integer array file (keeping in mind the naming convention for formatted and unformatted integer array files as documented in Section 2.10 of Part A of this manual). The "generate" files which GRID2ARC produces only record cells for which array elements within this integer array are non-zero. In most cases the window integer array will be a layer activity array.

Next GRID2ARC prompts for the names of the "line" and "points" files which it must write:

```
 Enter name for output "line" file:
 Enter name for output "points" file:
```

Part of a "line" file is shown below:

```
     10
    410824.094,        7260109.351
    411026.771,        7259888.168
    410805.588,        7259685.491
    410602.910,        7259906.674
    410824.094,        7260109.351
 end
     11
    411026.771,        7259888.168
    411229.448,        7259666.985
    411008.265,        7259464.308
    410805.587,        7259685.491
    411026.771,        7259888.168
 end
     12
    411229.448,        7259666.985
    411432.125,        7259445.802
    411210.941,        7259243.125
    411008.264,        7259464.308
    411229.448,        7259666.985
```

**Fragment of a "line" file written by GRID2ARC.**

Within a GRID2ARC-generated "line" file each cell of the finite-difference grid is represented by 7 lines of data. The first line contains the cell label; this is the cell number as obtained by counting cells row by row from the top left corner of the finite-difference grid. Next follow the cell corner coordinates, easting then northing; the first set of coordinates is repeated for polygon closure. Finally the "end" string signals that all information pertaining to a particular polygon has been supplied.

Part of a GRID2ARC-generated "points" file is shown below:

```
     10,        410814.841,        7259897.421
     11,        411017.518,        7259676.238
     12,        411220.195,        7259455.055
     13,        411422.872,        7259233.872
     23,        413449.643,        7257022.040
     24,        413652.319,        7256800.856
     25,        413854.997,        7256579.673
     26,        414057.673,        7256358.490
     27,        414260.351,        7256137.307
     28,        414463.028,        7255916.124
     29,        414665.705,        7255694.940
     42,        410593.657,        7259694.744
     43,        410796.334,        7259473.561
     44,        410999.011,        7259252.378
     45,        411201.688,        7259031.194
```

**Fragment of a "points" file written by GRID2ARC.**

Each line of a "points" file pertains to a single cell of the finite-difference grid. It contains three entries, viz. a cell's number (see above) and the easting and northing of its centroid.

Collectively the "line" and "points" files allow ARCINFO to generate a set of polygons, each polygon representing one cell of the finite difference grid. Within ARCINFO each of these polygons can be assigned attributes on the basis of other geographical information covering the same model area.

**Uses of GRID2ARC**

"Line" and "points" files produced by GRID2ARC can be imported into ARCINFO. Once imported, the active part of a model layer can be represented within ARCINFO as a set of polygons, one for each cell. This representation allows ARCINFO to undertake a large number of important pre- and postprocessing operations pertaining to a particular model. For example a data column containing integers or real numbers can be added to the cell number table, integer or real values within this column being assigned on the basis of spatial relationships between grid cells and components of other coverages pertaining to the same area. A table so constructed can then be exported from ARCINFO as an ASCII file and converted to a three-column integer or real array table file using program TABCONV. The table can then be reformatted as a real or integer array, ready for model usage, by program TAB2INT or TAB2REAL. Conversely an existing real or integer array can be imported into ARCINFO using REAL2MIF or INT2MIF to generate a real or integer array table (use the "MID" file written by either of these programs) followed by TABCONV to undertake the conversion from the use of row and column number format to cell number format.

More sophisticated analysis and preprocessing of model data can also be carried out within ARCINFO. For example model cell polygons can be intersected with soil and land use polygons to determine the area of each soil type/land use combination within each cell. This information can then be combined with the leaching fraction pertinent to each soil type/land use combination to calculate a recharge array for a model.

**See Also**

See also INT2MIF, REAL2MIF, TAB2INT, TAB2REAL, TABCONV.

# GRID2BLN

**Function of GRID2BLN**

Program GRID2BLN produces a SURFER blanking (ie. "XYLine") file of part or all of the finite difference grid. This file can be used by SURFER to draw a picture of the model grid to overlay on basemaps and/or contour plots.

**Using GRID2BLN**

A settings file `settings.fig` must be present in the directory from which GRID2BLN is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution GRID2BLN prompts:

```
 Enter name of grid specification file:
```

Type in the name of a grid specification file. Alternatively, if a filename file (`files.fig`) is present in the current directory, GRID2BLN may provide a default filename with the above prompt; press <Enter> to accept the default or type in the required name. GRID2BLN reads from the grid specification file the geographical information it requires in order to construct a SURFER blanking file.

GRID2BLN next prompts:

```
 Enter name of window integer array file:
```

Here you should supply the name of a formatted or unformatted file holding a MODFLOW/MT3D-compatible integer array (see Section 2.10 of Part A of this manual). For the purposes of constructing the SURFER blanking file, cells with non-zero integer array elements are deemed as "active", zero-valued cells being considered "inactive". Thus if a MODFLOW layer activity array is used as a window integer array, SURFER will draw a map of the active part of the finite-difference grid (for that layer), omitting all inactive cells.

Next GRID2BLN asks the user to supply a name for the blanking file which it is about to write:

```
 Enter name of SURFER blanking output file:
```

Here enter a filename of your choice. However it is recommended that you provide this filename with an extension of "`BLN`" so that SURFER will recognise it as a blanking file.

It is important to note that SURFER has two uses for blanking files. They can be used simply to draw a picture, or they can be used to actually blank part of a contouring grid. While blanking files produced by GRID2BLN are suitable for drawing the finite-difference grid, they are not suitable for blanking the SURFER contouring grid

as the elements represented in a GRID2BLN-generated blanking file are not polygons; they are simply line segments. To produce a file suitable for blanking, use program ZONE2BLN.

**Uses of GRID2BLN**

GRID2BLN can be used in preparing diagrams of the model grid in which the latter is plotted in real-world coordinates. When superimposed on other maps of a study area, a clear picture emerges of the spatial relationships between the groundwater model and cadastral, topographic, geological, and other data.

**See Also**

See also programs GRID2DXF and  ZONE2BLN.

# GRID2DXF

**Function of GRID2DXF**

Program GRID2DXF writes a DXF file of all or part of the finite-difference grid. This file can be used by CAD, plotting, contouring and GIS software to draw a picture of the finite difference grid, superimposed on other geographical information.

**Using GRID2DXF**

A settings file `settings.fig` must be present in the directory from which GRID2DXF is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution GRID2DXF prompts:

```
Enter name of grid specification file:
```

Type in the name of a grid specification file. Alternatively, if a filename file `files.fig` is present in the current directory, GRID2DXF may include a default grid specification filename with the above prompt; press <Enter> to accept the default or type in the correct name. GRID2DXF reads from the grid specification file the geographical information it requires in order to represent the grid in DXF format.

GRID2DXF next prompts:

```
Enter name of window integer array file:
```

Here you should supply the name of a formatted or unformatted file holding a MODFLOW/MT3D-compatible integer array (see Section 2.10 of Part A of this manual). For the purposes of constructing its DXF output file, cells with non-zero integer array element are deemed "active", zero-valued cells being considered "inactive". Thus if a MODFLOW layer activity array is used as a window integer array, GRID2DXF will write a DXF file in which only the active cells of the finite-difference grid (for that layer) are represented.

GRID2DXF's next prompt is:

```
Enter name for DXF output file:
```

Here supply a filename of your choice; however it is a good idea to provide a filename extension of "DXF" so that the file is easily recognised as a DXF file.

**Uses of GRID2DXF**

GRID2DXF is used in preparing maps which include the model grid. When the model grid is superimposed on other maps of a study area, a clear picture emerges of the

spatial relationships between the groundwater model and cadastral, topographic, geological, and other data.

**See Also**

See also GRID2BLN, ZONE2DXF.

# GRID2PT

**Function of GRID2PT**

Program GRID2PT tabulates the coordinates of the cell centres of the "active" part of the finite-difference grid. The "active" part of the grid is defined by non-zero values within a user-supplied window integer array. If certain GRID2PT options are chosen, GRID2PT can produce a bore coordinates file in which each "bore" is an active grid cell centre. This can then be used with other Groundwater Data Utilities to accomplish such tasks as determining model-calculated results at the centres of certain user-specified cells.

**Using GRID2PT**

A settings file `settings.fig` must be present in the directory from which GRID2PT is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

Upon commencement of execution GRID2PT prompts:

```
 Enter name of grid specification file:
```

Type in the name of the grid specification file. Alternatively, if a filename file (`files.fig`) is present within the current directory, GRID2PT may include a default filename with the above prompt; press <Enter> to accept the default or type in the name of the grid specification file that you would prefer to use. GRID2PT reads from the grid specification file the geographical information which it requires in order to determine the centroid coordinates of grid cells.

GRID2PT then prompts:

```
 Enter name of window integer array file:
```

Here supply the name of a formatted or unformatted file holding a MODFLOW/MT3D-compatible integer array (see Section 2.10 of Part A of this manual). Cells with an integer value of zero within this array are considered to be "inactive"; the coordinates of such "inactive" cells are not represented in the GRID2PT output file.

GRID2PT's next prompt is:

```
 Enter name for output file:
```

Here supply a filename of your choice to which GRID2PT will write model grid cell centre coordinates. GRID2PT then asks the user for his/her wishes regarding presentation of data on this output file. First:

```
 In GRID2PT output file:-
```

```
 Use row and column numbers or cell numbers?  [r/c]
```

GRID2PT can use either of two methods to identify a cell. The first is the traditional method of row and column numbers (using the MODFLOW convention whereby the cell with a row and column number of 1 occupies the top left corner of the grid). The second method characterises each cell by a single number instead of two; this single number is referred to as the "cell number". A cell's "cell number" is obtained by counting cells row by row starting at the top left corner of the grid. If you wish to use GRID2PT to construct a pseudo bore coordinates file you should select the cell number option for cell identification, this single number acting as the bore identifier in the pseudo bore coordinates file.

Next GRID2PT asks:

```
 Include dummy layer number column?  [y/n]
```

If you respond with "y" GRID2PT will add an extra column to its output file. This column will contain the number that you supply in response to the following GRID2PT prompt:

```
 Enter dummy layer number:
```

Using this option together with the cell number option allows GRID2PT to generate a valid bore coordinates file. The latter can then be used in conjunction with programs such as MOD2OBS in order to retrieve model-generated heads at specific cell centres.

The following example shows an extract from a file written by GRID2PT in which the cell number and dummy layer number options were selected. The dummy layer number was entered as 1.

```
        2724        434629.899       7248389.057     1
        2725        435023.905       7248696.887     1
        2726        435417.910       7249004.718     1
        2727        435811.914       7249312.549     1
        2728        436205.920       7249620.379     1
        2729        436599.926       7249928.210     1
        2730        436993.932       7250236.042     1
        2731        437387.936       7250543.872     1
        2732        437781.942       7250851.703     1
        2733        438175.948       7251159.534     1
        2734        438569.953       7251467.364     1
        2735        438963.957       7251775.195     1
        2736        439357.963       7252083.026     1
        2737        439751.969       7252390.857     1
        2738        440145.975       7252698.687     1
        2739        440539.979       7253006.518     1
        2740        440933.985       7253314.349     1
        2741        441327.990       7253622.179     1
        2742        441721.996       7253930.010     1
        2743        442116.000       7254237.841     1
        2744        442510.006       7254545.671     1
        2745        442904.012       7254853.502     1
        2746        443298.016       7255161.333     1
```

**Extract from a GRID2PT output file.**

**Uses of GRID2PT**

The primary use of GRID2PT is to produce a bore coordinates file based on certain user-identified cells. Target cells can be selected using a graphical MODFLOW preprocessor, a GIS, or any software from which an integer array can be generated and exported; all elements in this array should be zero except for the elements corresponding to the cells of interest.

Once a bore coordinates file has been constructed based on the selected cells, programs within the Groundwater Data Utilities can be used to process the cells as if they were bores. Thus MOD2OBS can be used to determine heads at the identified cells at one or more times during a model run. A bore sample file, based on a model run, can be created for the cell centres using program MOD2SMP. In the latter case program SMP2HYD can then be used to plot bore hydrographs, and program SMP2INFO can be used to temporally interpolate model-calculated quantities for these cells to times that do not correspond to model output times.

**See Also**

See also PTINGRID.

# INT2MIF

**Function of INT2MIF**

INT2MIF generates a MAPINFO-compatible MIF/MID file pair holding the geographical and array information contained in a MODFLOW/MT3D-compatible integer array. The files generated by INT2MIF can be used by MAPINFO (and other geographical information systems) to import a model integer array for GIS-based model pre/postprocessing and display.

**Using INT2MIF**

A settings file `settings.fig` must be present in the directory from which INT2MIF is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution INT2MIF prompts:

```
 Enter name of grid specification file:
```

Type in the name of the grid specification file pertinent to the current model. Note that if a filename file (`files.fig`) resides in the directory from which INT2MIF is run, the name of a default grid specification file may appear as part of the above prompt; in this case press <Enter> to accept the default or enter an alternative filename.

INT2MIF reads two integer arrays. First it asks for a "data" integer array. This is the integer array whose contents are to be written to the MIF and MID files for subsequent uploading into a GIS. INT2MIF prompts:

```
 Enter name of data integer array:
```

to which you should respond with an appropriate filename, taking note of the convention for integer array filename extensions employed by the Groundwater Data Utilities; see Section 2.10 of Part A of this manual. Next INT2MIF prompts:

```
 Enter name of window integer array file:
```

The window integer array file acts as a mask or "cookie-cutter" over the data integer array. Only cells of the latter array whose corresponding cells in the former array are non-zero-valued appear in the MIF/MID files written by INT2MIF. Note that the data and window integer array files can be the same.

INT2MIF next requests the names of the files to which it should write the geographical and array information respectively pertaining to the active window (as defined by the window integer array) of the data integer array. INT2MIF prompts:

```
 Enter name for output "MIF" file:
 Enter name for output "MID" file:
```

Then it asks:

```
 Enter AMG zone number of model area  [47 - 58]:
```

in response to which the Australian Map Grid zone number of the current study area should be entered. (If you do not live in Australia provide a zone number within the range indicated. Then, after INT2MIF has finished execution, you should replace the third line of the MIF file generated by INT2MIF with more appropriate `Coordsys` information obtained from file `MAPINFOW.PRJ` supplied with MAPINFO.)

INT2MIF then calculates the coordinates of the corners of each "active" cell and transfers them in appropriate format, to the MIF file whose name was supplied above. The contents of the "active" cells of the data integer array are written to the user-nominated MID file.

**Uses of INT2MIF**

INT2MIF provides the means whereby integer array data can be imported into a geographical information system (GIS). In a GIS grid cells can be displayed as "regions" or "polygons", with attributes assigned to each such cell. These attributes, uploaded through the MID file generated by INT2MIF, are the row and column numbers of each "active" cell, and the integer array value for each such cell. The latter can be schematised (for example as a thematic map) and overlain on other geographical information and/or images covering the study area. Within the GIS integer array values can be edited with reference to other information layers; row and column numbers should not be edited. The modified integer array can then be downloaded as an integer array table, and rewritten in integer array format using program TAB2INT.

**See Also**

See also REAL2MIF, TAB2INT, TAB2REAL.

# INT2REAL

**Function of INT2REAL**

INT2EAL constructs or modifies a MODFLOW/MT3D-compatible real array on the basis of a MODFLOW/MT3D-compatible integer array. Real array elements are assigned values on the basis of a user-supplied correspondence between integers as represented in the integer array, and real numbers to be written to the real array.

**Using INT2REAL**

A settings file `settings.fig` must be present in the directory from which INT2REAL is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution, INT2REAL prompts:

```
 Enter name of grid specification file:
```

Supply an appropriate filename. If a default name for the grid specification file appears with the above prompt (INT2REAL having obtained the default filename from a filename file residing in the current directory) press <Enter> to accept it or enter the more appropriate filename as described above.

INT2REAL next prompts the user for the name of an integer array file upon which to base its construction or modification of a model-compatible real array:

```
 Enter name of integer array file:
```

Supply an appropriate filename, keeping in mind the naming conventions for integer arrays outlined in Section 2.10 of Part A of this manual.

INT2REAL can either modify a real array or create one. In the former case it must read an existing real array before it can perform the modification. It prompts:

```
 Modify an existing real array or create a new one?  [m/c]:
```

Enter "m" or "c" as appropriate. If you enter "m", INT2REAL next prompts for the file which holds the real array to be modified:

```
 Enter name of file holding existing real array:
```

to which you should respond with an appropriate filename, keeping in mind the naming convention for real array files outlined in Section 2.17 of Part A of this manual. The real array is then read by INT2REAL and stored within memory.

INT2REAL next examines the integer array which it has previously read, making an internal list of all integers present in that array. If the option to create a new real array was selected then a real number must be supplied corresponding to every integer

occurring in this list. Alternatively, if an existing real array is to be modified, real numbers need to be provided for only some of these integers. In the former case INT2REAL assigns a value to every cell in the new real array on the basis of the integer found in the corresponding cell of the integer array and on the user-supplied linkages between integers and real numbers. In the latter case, the integer-real linkage is used to provide new values for real array cells only where the user has actually provided a real number corresponding to a given integer.

The correspondence between integers and real numbers can be supplied to INT2REAL from the terminal in response to screen prompts, or it can be supplied through a file. So INT2REAL next asks how these correspondences are to be entered:

```
Enter integer-real correspondence manually or using a file? [m/f]:
```

If you enter "f", INT2REAL requests the name of the file in which the correspondences are recorded:

```
Enter name of integer-real correspondence file:
```

An integer-real correspondence file is shown below.

```
0   3.445
1   3.6e-3
2   45.54
3   5.445
4   9.586
5   545.6
```

**An integer-real correspondence file.**

An integer-real correspondence file is comprised of two columns of data, the first column holding integers and the second column holding real numbers. If a real array is to be created by INT2REAL, every integer cited in the integer array must also be cited in the integer-real correspondence file. If an existing real array is to be modified, only some of the integers found in the integer array need to be listed in the integer-real correspondence file. Note that the same real number can correspond to more than one integer.

If you inform INT2REAL that integer-real correspondences are to be supplied from the terminal, then it does not prompt for the name of an integer-real correspondence file. Instead it presents the user with each of the integers that it found in the integer array and requests that he/she supply a corresponding real number. If a real array is being created (rather than modified) a real number must be supplied for every integer. However if a real array is being modified, you may press the <Enter> key in response to INT2REAL's request for a real number corresponding to a certain integer, thus signalling that real array elements whose row and column numbers are identical to those containing that integer in the integer array are not to be modified. INT2REAL's prompts are:

```
The following integers have been detected in the integer array:-
Enter corresponding real numbers.
Press <ENTER> if integer effects no change to existing real array.
  Enter real number corresponding to integer 0:
  Enter real number corresponding to integer 1:
```

```
   Enter real number corresponding to integer 2:
                    etc.
```

Once integer-real correspondences have been supplied, either from a file or from the terminal, INT2REAL prompts for the name of the file to which it should write its created or modified real array:

```
 Enter name for output real array file:
```

in response to which an appropriate filename should be supplied, keeping in mind the naming conventions for real arrays outlined in Section 2.17 of Part A of this manual.

**Uses of INT2REAL**

INT2REAL provides the means whereby aquifer property, recharge, or other data can be supplied to a model on the basis of a zonation pattern established over the model area. The zonation may be derived from geological, land use, etc. data. It may have been created within a model preprocessor or within a GIS such as MAPINFO; in either case the zonation pattern is supplied to INT2REAL in the form of an integer array.

If a zonation-defining integer array is based on model area land use, it may serve as a basis for recharge array construction. Different recharge arrays can be built for different model stress periods and different projected land-use scenarios, all on the basis of a single integer array, a different integer-real correspondence file being employed for each occasion. Such real arrays can be created "from scratch" or modified (using the INT2REAL "modify" option) from a regional recharge array as different land-use scenarios are tested.

Program INT2REAL can also form a valuable component of a "composite model" for which parameters are estimated using PEST. See Section 4.14 of Part A of this manual for details.

**See Also**

See also REAL2INT.

# LAYDIFF

**Function of LAYDIFF**

LAYDIFF reads data from a bore sample file. In the following discussion it will be assumed that head data is read; however LAYDIFF's operations are not restricted to only data of this type. It also reads bore eastings, northings and layer numbers from a bore coordinates file. For a particular user-supplied day, LAYDIFF evaluates head differences between model layers at locations where this is possible by subtracting the head in one layer from that in another layer. This calculation can only be performed if two conditions are met. The first condition is that the bores which tap the two different layers are separated by a horizontal distance that is not very large (this being defined by a user-supplied threshold). The second condition is that at least one sample from each of the bores upon which head difference calculation is based was taken within a user-supplied time window about the specified time at which head difference calculations are required. Where borehole head samples bracket this specified time, linear interpolation between those samples to the specified "head difference" reference time is undertaken by LAYDIFF.

**Using LAYDIFF**

Like many other members of the Groundwater Data Utility Suite, LAYDIFF requires the presence of a settings file `settings.fig` in the directory from which it is run. The contents of this file inform LAYDIFF of the protocol to use for representation of dates. An optional filename file (`files.fig`), informing LAYDIFF of the names of a default bore coordinates file and a default bore sample file pertaining to its current task, may also be present within the current working directory.

LAYDIFF commences execution with the prompt:-

```
  Enter name of bore coordinates file:
```

Supply the name of the appropriate file (or accept the default provided by LAYDIFF by simply pressing the <Enter> key). Note that the fourth column of this file must contain layer numbers. Note also that a subset of bores provided in this file can be selected for processing using a listing file whose name is provided in response to LAYDIFF's second prompt:-

```
  Enter name of bore listing file:
```

The bore listing file can be the same as the bore coordinates file if desired.

LAYDIFF prompts for the name of the bore sample file in which borehole measurements are housed:-

```
  Enter name of bore sample file:
```

As usual, ensure the integrity of this file by checking it with SMPCHEK before supplying it to LAYDIFF.

LAYDIFF next prompts for the date and time at which inter-layer head differences are to be calculated:-

```
Enter reference date [mm/dd/yyyy]:
Enter reference time [hh:mm:ss]:
```

Where necessary, LAYDIFF performs linear interpolation of measurements residing in the bore sample file to this exact date and time. However no temporal interpolation will take place (and hence data pertaining to the bore will be ignored) if the time pertaining to the nearest sample for that bore is outside of a given time window, the width of which is specified in response to the prompt:-

```
Enter maximum days to reference date (fractional if necessary):
```

Note that if the reference date/time is not subtended by two different samples, no linear interpolation can take place to that date/time. In this case LAYDIFF uses the head at the nearest sample time as the head at the reference date and time, provided the time difference between the measurement and reference times does not exceed the above time difference threshold.

As well as a temporal threshold, a distance threshold is also required for interlayer head difference calculation. For any bore situated within a particular model layer, LAYDIFF finds the closest bore in all underlying layers in order to evaluate interlayer head differences between measurements in the pertinent bores. However unless the horizontal distance between the upper layer bore and the lower layer bore is less than a given threshold, no inter-layer head difference calculation takes place between those bores. This distance threshold is supplied to LAYDIFF following the prompt:-

```
Enter exclusion distance:
```

Once this question has been answered, LAYDIFF prompts for the name of the file to which to write its calculated head differences. It then proceeds to write this file.

Finally LAYDIFF prompts:-

```
Generate an instruction file to read output file? [y/n]:
```

Type "y" or "n" as appropriate. If you type "y", LAYDIFF will write a file (named according to the user's choice) containing an instruction set to read inter-layer head differences written to the LAYDIFF output file. In this instruction file, observations are named according to the convention "*bore1-bore2*" where "bore1" is the name of the upper layer bore and "bore2" is the name of the lower layer bore involved in each head difference calculation. LAYDIFF does not observe the 12 character limit on PEST observation names in formulating the above observation name. However if any observation name exceeds 12 characters in length it warns the user of this, leaving it up to him/her to then shorten its name in the most appropriate way.

Note the following aspects of LAYDIFF's calculations:-

1.  LAYDIFF attempts to use every bore in the listing file as a reference point for calculation of interlayer head differences. However if, for a particular bore, there is no measurement close enough to the reference date and time (as defined above), that bore is ignored and no head differences are calculated at that site.

2.  For each bore, head differences are calculated only to underlying layers (ie. to layers with greater layer number). Multiple differences, pertaining to multiple underlying layers, are calculated where bore locations permit.

3.  If, for any upper layer bore, there is no bore in any underlying layer which is closer to that bore than the user-supplied exclusion distance, the upper layer bore is ignored.

**Uses of LAYDIFF**

LAYDIFF can be used on its own or as part of a model calibration exercise. The inclusion of inter-layer head differences in the calibration dataset can often be of great use in the estimation of vertical conductivities and/or inter-layer conductances. LAYDIFF allows head differences to be calculated both for field data and for model-generated datasets; in the latter case MOD2SMP should be run prior to LAYDIFF, this providing model-generated heads in bore sample file format. If field data resident in a measurement bore sample file are then interpolated to this model-generated bore sample file (using SMP2SMP), equivalent field and model-generated bore sample files will be available. LAYDIFF can then run on each of these; head differences calculated from the field bore sample file can be transferred to the PEST control file as the "observed" set of head differences, while those generated from the model bore sample file can be used as the model-generated counterpart to these. These latter differences can, of course, be read using the LAYDIFF-generated instruction set.

**See Also**

See also MOD2SMP, SMP2SMP.

# MANY2ONE

**Function of MANY2ONE**

MANY2ONE reads an unformatted MODFLOW or MT3D output file containing a series of two-dimensional arrays, each array pertaining to a different layer, time and, possibly, transport step. MODFLOW writes such lengthy, unformatted files to record head, drawdown, subsidence, etc data, while MT3D records its calculated concentrations to such files. MANY2ONE presents the user with the contents of each array (as read from MODFLOW or MT3D supplied headers) and gives him/her the option of storing a particular array in a separate file in either formatted or unformatted form.

**Using MANY2ONE**

A settings file `settings.fig` must be present in the directory from which MANY2ONE is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

Upon commencement of execution MANY2ONE prompts:

```
Enter name of MODFLOW/MT3D unformatted output file:
```

to which you should respond with an appropriate filename. MANY2ONE next needs to know whether the multiple-array file has been generated by MODFLOW or MT3D. The distinction is necessary as these programs write different unformatted array headers to their output files; MANY2ONE must read these headers so that it can convey the information contained in them to the user. Hence it prompts:

```
Is this a MODFLOW or MT3D output file?  [f/t]:
```

to which you should reply with "f" or "t" as appropriate.

MANY2ONE then reads each array in the MODFLOW or MT3D output file, writing a description of the contents of the array to the screen:

```
MODFLOW head array for layer 1 ----->
 Stress period                            = 1
 Time step                                = 1
 Elapsed time since start of stress period  = 30.00000
 Elapsed time since start of simulation     = 30.00000
 Store array in separate file?  [y/n]:
```

Indicate using the "y" or "n" keys whether MANY2ONE should store the array in a separate file or not. If "y", MANY2ONE prompts:

```
Enter name for real array file:
```

to which an appropriate filename should be entered, keeping in mind the naming convention for real arrays outlined in Section 2.17 of Part A of this manual.

If the array is stored in unformatted form (ie. if an extension of "REU" is supplied with the filename or unformatted storage is explicitly requested), MANY2ONE records the array header information in the new, unformatted file. This allows the file to serve as an input file for MODBORE or MT3BORE from the PEST MODFLOW/MT3D Utilities suite which can then interpolate its contents to user-specified boresites.

**Uses of MANY2ONE**

Arrays extracted from large MODFLOW or MT3D output files can be written in formatted form for user inspection, either on the screen or after printing. Contour maps can be produced from individual arrays using, for example, program REAL2SRF. Such arrays can also serve as inputs to program SECTION which is able to plot transects of arbitrary complexity through a model grid. An array extracted from a MODFLOW or MT3D unformatted output file can be re-used as an initial conditions array, zoned to form an integer array, imported into a GIS, imported into a model graphical preprocessor, etc. Hence MANY2ONE serves as an important link between model-generated data and many of the utilities documented in this manual.

# MOD2OBS

**Function of MOD2OBS**

Program MOD2OBS generates a bore sample file on the basis of results stored in an unformatted MODFLOW or MT3D output file. However unlike program MOD2SMP which generates "samples" at model output times, MOD2OBS generates "samples" at the same dates and times as samples recorded in an existing bore sample file. It achieves this by carrying out spatial interpolation to the sites of bores, and temporal interpolation of model results to measurement dates and times. Thus MOD2OBS provides a means of directly comparing field data with model-generated data. It can thus form a vital component of a composite model used in a calibration setting.

**Using MOD2OBS**

MOD2OBS commences execution with the prompt:-

```
 Enter name of grid specification file:
```

Type the name of a grid specification file. Alternatively, if a filename file (`files.fig`) is present in the current directory, MOD2OBS may provide a default filename with the above prompt; press <Enter> to accept the default or supply the required name.

Next MOD2OBS prompts for the name of a bore coordinates file:-

```
 Enter name of bore coordinates file:
```

Once again, if a filename file is present in the current directory a default may be supplied; this can be accepted simply by pressing the <Enter> key. MOD2OBS requires bore coordinates so that it can spatially interpolate model results from grid cell centres to the sites of observation bores. The user is able to select which bores will be involved in the interpolation process by providing the name of a bore listing file in response to the prompt:-

```
 Enter name of bore listing file:
```

Each bore cited in the bore listing file should also be cited in the bore coordinates file. If desired, the bore listing file can also be the bore coordinates file; thus all bores cited in the bore coordinates file will take part in the interpolation process.

Next MOD2OBS prompts for the name of a bore sample file:-

```
 Enter name of bore sample file:
```

As usual, if the name a bore sample file appears in the filename file (`files.fig`) situated in the current directory, that name will appear as a default; it can be accepted by simply pressing the <Enter> key.

While the bore sample file can contain data pertaining to many more bores than those listed in the bore listing file, and record sample values over a time interval far exceeding the model simulation time, it is a good idea to reduce the amount of redundant information present in that file, as far as MOD2OBS's present task is concerned, to a minimum. This is because MOD2OBS must allocate sufficient memory to hold virtually all of the information contained within this file as it executes; thus unless your machine possesses a lot of RAM, allocation problems may ensue. Should this occur, MOD2OBS will issue an appropriate error message and terminate execution.

It is MOD2OBS's task to spatially interpolate model results to the locations of bores cited within the bore sample file (and bore listing file), and to undertake time-interpolation of model results to the times at which samples were taken for each such bore. Thus, for each sample within the user-provided bore sample file for which it is possible to undertake time-interpolation from within the model simulation timespan, MOD2OBS will provide a corresponding model-generated "sample". The set of such samples will be recorded in bore sample file format.

MOD2OBS reads model results from a MODFLOW or MT3D output file of the same type as is used to store heads, drawdowns and concentration data (as well as other array data such as compaction in certain MODFLOW packages). In these files, model array data is recorded layer by layer. MOD2OBS prompts the user for the name of the unformatted file which it must read to obtain this model-calculated data:-

```
 Enter name of unformatted model-generated file:
```

It is the user's responsibility to ensure that MODFLOW or MT3D stores enough data in this file to allow accurate temporal interpolation to bore sample times. For times which do not correspond to model output times, MOD2OBS performs a linear interpolation between arrays present in the unformatted model output file. Hence, through the setting of appropriate Output Control variables, the user should ensure that unformatted output is provided at close enough time intervals for linear interpolation to be accurate. Furthermore, data should be provided for all model layers that contain at least one listed bore. As spatial interpolation is intra-layer only (employing a bilinear interpolation scheme), it is not necessary that the unformatted model output file contain data for any other model layers.

MOD2OBS needs to know whether it is dealing with a MODFLOW or MT3D unformatted output file. This is because the header to each unformatted array is slightly different for each of these models. So it prompts:-

```
 Is this a MODFLOW or MT3D file?  [f/t]:
```

It also needs to know the "threshold value", above which a cell is considered to be inactive or dry:-

```
 Enter inactive threshold value for arrays in this file:
```

MOD2OBS then asks a series of questions, the answers to which will allow it to calculate the date and time corresponding to each model output time, the latter being recorded in the header to each unformatted array:-

```
Enter time units used by model (yr/day/hr/min/sec) [y/d/h/m/s]:
Enter simulation starting date [dd/mm/yyyy]:
Enter simulation starting time [hh:mm:ss]:
```

(Note that the date format used by MOD2OBS depends on the contents of file `settings.fig` situated within the current directory. If this file is not present, MOD2OBS will not run.)

MOD2OBS then asks:-

```
 How many layers in the model?
```

It needs to know this so that it can dimension arrays appropriately before embarking on its calculations.

MOD2OBS then prompts:-

```
If a sample time does not lie between model output times, or if there
is only one model output time, value at the sample time can equal
that at nearest model output time:-
Enter extrapolation limit in days (fractional if necessary):
```

If a sample from a particular bore lies either before the first time of model output, or after the last time of model output, MOD2OBS cannot perform a linear interpolation from model output times to the bore sample time. Hence it will calculate an "extrapolated" value at that time equal to the first or last model-calculated value respectively for that bore. However, there is a limit to the time over which such extrapolation can take place; this is set by the user's response to the above prompt.

Note that the earliest time at which MODFLOW output is available is the end of the first time step. Hence if you require model output at a time as close as possible to the beginning of the simulation, make this first time step as short as possible.

Next MOD2OBS requests the name of the bore sample file which it must write:-

```
 Enter name of bore sample output file:
```

MOD2OBS carries out temporal and spatial interpolation to the sites and times cited in the user-provided bore sample file. It then produces a bore sample file of its own, with samples at exactly the same dates and times as those occurring within the user-provided bore sample file, but with model-generated numbers substituted for measured ones. It is important to note, however, that there may not be a sample in the MOD2OBS-generated bore sample file corresponding to *every* sample in the user-provided bore sample file. MOD2OBS does not generate a sample for a particular bore and time under the following conditions:-

- if a user-supplied sample precedes the earliest time of model output by an amount exceeding the extrapolation limit,

- if a user-supplied sample postdates the latest time of model output by an amount exceeding the extrapolation limit,

- if there are no arrays within the unformatted model output file for the layer holding a listed bore,

- if a bore appears in the bore listing file but not in the bore sample file,

- if a bore appears in the bore sample file but not in the bore listing file.

Interpolation also cannot take place if a bore does not lie within the finite-difference grid, or if it lies within an inactive or dry cell; under such circumstances an appropriate message replaces the sample value in the MOD2OBS-generated bore sample file. Thus if a cell becomes dry during a PEST run in which MOD2OBS forms part of a composite model, the same instruction set can read the bore sample file generated after all model runs because there will be no difference in the number of bores listed after any run. (However if the instruction set includes an instruction to read the head value for a bore in the dry cell, an error condition will arise. The resulting PEST error message will be such as to direct the user to the source of the problem.)

**Uses of MOD2OBS**

MOD2OBS finds its greatest use in model calibration using PEST. A comparison between observed borehole data and its model-generated counterparts is easily achieved by running MODFLOW/MT3D followed by MOD2OBS as a composite model. Because MOD2OBS performs both spatial and temporal interpolation to the sites and times of measured data, and presents the results of its calculations in the same format as the measured data (ie. as a bore sample file), comparison between the two datasets can be made with ease. When used in conjunction with PEST (with the aid of program PESTPREP for PEST input file generation), MOD2OBS forms a vital component of the model calibration process.

MOD2OBS is just as useful in steady-state MODFLOW calibration as it is in transient calibration. However in this case there will normally be only one "time" at which there is model output (equal to the notional "elapsed simulation time" since the beginning of the simulation). In this case MOD2OBS cannot interpolate between neighbouring MODFLOW output times to the time of a bore reading. Instead, using the extrapolation facility discussed above, it conducts only spatial interpolation to sample sites, assuming temporal coincidence of borehole sample times with model output times. In this case the user should ensure that the measurement bore sample file contains steady-state samples which are all referenced to a date and time which is close to the notional model output time (ie. within the temporal extrapolation limit). This is a simple matter if it can be assumed that steady-state conditions prevail on a certain date. Beware, however, of making the user-supplied extrapolation time too large, for then neighbouring samples in the user-supplied bore sample file may be close enough to the notional model output time to warrant inclusion in the MOD2OBS-generated bore sample file. If this occurs you must either decrease the extrapolation time, or assign duplicated observations a weight of zero.

**See Also**

See also MOD2DAT, PESTPREP and SMP2SMP.

# MOD2SMP

**Function of MOD2SMP**

MOD2SMP reads an unformatted output file generated by MODFLOW or MT3D. If written by MODFLOW this file may contain layer-specific arrays of head, drawdown, compaction, preconsolidated head or subsidence at one or a number of elapsed simulation times. An unformatted MT3D output file may contain concentration data expressed in similar format. MOD2SMP interpolates the data contained within these arrays to a user-specified set of boresites at all recorded model output times, writing its output in the form of a bore sample file. Thus those programs documented in this manual which are able to read and manipulate field data stored in a bore sample file can process model-generated data in the same way.

**Running MOD2SMP**

Program MOD2SMP will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities.

MOD2SMP commences execution with the prompt:

```
 Enter name of grid specification file:
```

If the name of a grid specification file has been read from a filename file (`files.fig`) resident in the current directory, it will be included in the above prompt as a default filename. Either press <Enter> to accept the default or type in the appropriate name. MOD2SMP needs to know grid specifications so that it can carry out interpolation from the model grid to user-specified boresites. As coordinates must be supplied for the latter, MOD2SMP next requests the name of a bore coordinates file:

```
 Enter name of bore coordinates file:
```

(Depending on the existence and contents of a filename file in the current directory, a default filename may be included in the above prompt.) Note that the bore coordinates file must include the layer number to which each bore pertains; see Section 2.2 of Part A of this manual.

A user can select which of the bores in the bore coordinates file are to be represented in the MOD2SMP-generated bore sample file by listing the desired bores in a bore listing file. So MOD2SMP prompts:

```
 Enter name of bore listing file:
```

Note that the file provided previously as a bore coordinates file can be re-read as a bore listing file if desired; in this way all bores represented in the former file will be represented in the MOD2SMP-generated bore sample file.

MOD2SMP next prompts for the name of the unformatted MODFLOW or MT3D output file which it must read and interpolate to the bores cited in the bore listing file. This file should contain arrays for all layers containing bores for which information is required; obviously, to be of most use in generating a bore sample file, it should contain output arrays at more than one elapsed simulation time. MOD2SMP prompts:

```
 Enter name of unformatted model-generated file:
 Is this a MODFLOW or MT3D file?  [f/t]:
```

Then it asks:

```
 How many different output times are represented in this file?
```

MOD2SMP needs to know the number of different output times so that it can dimension arrays appropriately before reading the MODFLOW/MT3D-generated output file and carrying out spatial interpolation. Note that, for a multilayered model, there will be more arrays in the output file than there are model output times; it is the latter quantity that is required by MOD2SMP (unlike MODBORE from the PEST MODFLOW/MT3D Utilities which requires as part of its input dataset the number of actual arrays present in the MODFLOW or MT3D output file). Note also that if you are unsure of the contents of an unformatted MODFLOW or MT3D output file, you can use program MANY2ONE to read and report each of the array headers. Alternatively, supply a number that you are sure is greater than the number of output times; if it is not large enough MOD2SMP will soon inform you.

In order that it can adjust its spatial interpolation to take account of dry and inactive model cells MOD2SMP next prompts:

```
 Enter blanking threshold value for arrays in this file:
```

Enter a positive number that is less than HDRY and HNOFLO (if the array is a head or drawdown array generated by MODFLOW) or CINACT (if the array was generated by MT3D). Some MODFLOW preprocessors supply default values such as 999.99 and 1.0E30 for these variables. Others supply negative numbers as defaults; these are fine as long as their absolute value exceeds the above threshold.

Before it can generate a bore sample file, MOD2SMP needs to know how to convert elapsed model simulation times to true dates and times. So it asks:

```
 Enter time units used by model (yr/day/hr/min/sec) [y/d/h/m/s]:
 Enter simulation starting date [dd/mm/yyyy]:
 Enter simulation starting time [hh:mm:ss]:
```

(Note that the date format used in the second of the above prompts depends on the contents of the settings file settings.fig.) Then, after prompting:

```
 Enter name of bore sample output file:
```

MOD2SMP reads the MODFLOW/MT3D output file, interpolating the arrays contained therein to the sites of the bores listed in the bore listing file. MOD2SMP

uses an identical grid-to-point interpolation scheme to that of program MODBORE from the PEST MODFLOW/MT3D Utilities.

In its output bore sample file, MOD2SMP employs certain sample values to indicate certain error conditions; wherever these indicator values are employed, a fifth "x" column is added so that any of the Groundwater Data Utilities which read the file will ignore the dummy values. Thus a sample value of 7.1E37 indicates that a bore does not lie within the finite difference grid, a sample value of 5.1E37 indicates that the bore lies within a cell whose absolute model-generated value is above the blanking threshold and a value of 3.1E37 indicates that no arrays pertaining to the layer in which a bore lies were generated by MODFLOW or MT3D for a particular output time, in spite of the fact that arrays were generated for other model layers at this same output time.

**Uses of MOD2SMP**

Because it performs the dual functions of interpolating model results to bore locations and writing its results in the form of a bore sample file, use of MOD2SMP makes model-generated data "look like" field data. Thus those Groundwater Data Utilities which process the data contained in a bore sample file are able to carry out the same tasks on "model-generated field data" contained in a bore sample file written by MOD2SMP. For example program SMP2HYD can be used to construct data files that can be used by commercial plotting software to generate borehole hydrographs. These can be plotted on the same graphs as measured borehole hydrographs, thus facilitating a comparison between model-generated data and field data. Another possibility is that program SMP2DAT can be used to set up a PEST run based on "model-generated observations" housed in a MOD2SMP-generated bore sample file. In this way optimal PEST settings can be determined while carrying out an inversion for which the answer is known before applying PEST to model calibration based on real field data. Alternatively, a similar procedure can be carried out using MOD2OBS and PESTPREP and/or BUD2SMP, SMP2SMP and PESTPREP.

**See Also**

See also BUD2SMP, MODBORE, MOD2OBS, PESTPREP and SMP2SMP.

# PARM3D

**Function of PARM3D**

PARM3D facilitates the use of zones or pilot points in the parameterisation of a multi-layer model where hydrogeological units do not necessarily coincide with model layers. Parameterisation of a three-dimensional model domain is undertaken by assembling two-dimensional property arrays, or through using these arrays as a basis for vertical interpolation. These processes are guided by zonal dispositions, these also being assigned through two-dimensional integer arrays; if required, zonal disposition can be modified in the course of PARM3D execution.

When the parameterisation process is complete, PARM3D writes a series of two-dimensional property arrays (one for each model layer) to which MODFLOW's or MT3D's attention can be directed through appropriate OPEN/CLOSE statements inserted within its input files.

**Using PARM3D**

*Keyboard Input*

PARM3D differs from many of the programs of the PEST Groundwater Data Utilities in that keyboard input is minimal. PARM3D prompts for only one item of information, viz. the name of its control file. As is standard practice for members of the Groundwater Data Utilities, responding to this prompt by pressing the "e" key followed by the <Enter> key results in backtracking of program execution; in fact this results in termination of execution when undertaken in response to its first (and only) screen prompt.

Like most of the programs of the PEST Groundwater Data Utilities, PARM3D requires that a file named *settings.fig* exist in the directory from which it is run. PARM3D reads the status of the COLROW variable cited within this file. If this is supplied as "yes", then a "number of columns, number of rows" header is expected at the top of each formatted integer or real array which PARM3D reads; similarly, PARM3D includes this header as the first line of any real array file that it writes. Alternatively, if COLROW is supplied as "no", this header must not be present in any integer or real array file which PARM3D reads; nor will it be present in any array file which PARM3D writes.

*PARM3D Control File*

The PARM3D control file is subdivided into four sections. Optionally, as discussed below, one of these sections may be omitted. The names of these sections are:-

- control data

- layer files

- elevation files

- parameter value assignment

As is shown in the example below, each of these sections of the PARM3D control file must be introduced with a header stating the name of the section; in each case this name must follow an asterisk character.

```
# An example PARM3D input file.

* control data
"grid spec.spc"
6  1
* layer files
lay1.inf hcond1.ref
lay2.inf hcond2.ref
lay3.inf hcond3.ref
lay4.inf hcond4.ref
lay5.inf hcond5.ref
lay6.inf hcond6.ref
* elevation files
bot0.ref
bot1.ref
bot2.ref
bot3.ref
bot4.ref
bot5.ref
bot6.ref
* parameter value assignment
zone 1 33.00 overwrite
zone 3 99.99 overwrite
zone 2 grad_arith 2 topp2.ref bott2.ref overwrite
layer 1 rezone llay1.inf ignore_zero
layer 2 rezone llay2.inf ignore_zero
layer 3 rezone llay3.inf use_zero
zone 0 1000.0 overwrite
zone 9 5.55 overwrite
default -999
```

**Example of a PARM3D control file.**

A comment line can be inserted anywhere within a PARM3D control file; this line must begin with a "#" character. A blank line can also exist anywhere within the file.

Sections of the PARM3D control file are now discussed in detail.

*"Control Data" Section*

This section of the PARM3D control file contains only two lines. The first of these lines must contain the name of the grid specification file pertaining to the current model; see Part A of this manual for specifications of this file. If the name of this file contains a space, it must be enclosed in quotes.

The second line of the "control data" section must contain two integers. The first is the number of layers in the model (referred to as NLAY herein). The second is the value of a variable named ELEVFLAG. If ELEVFLAG is supplied as zero, then no layer elevation files will be provided later in the control file, and hence no "elevation files" section will be present within the current PARM3D control file. However if ELEVFLAG is provided with a non-zero value, layer elevation files must be cited within an "elevation files" section of the current PARM3D control file. ELEVFLAG

must not be set to zero if any of the "interp_arith", "interp_geom", "grad_arith" or "grad_geom" instructions are provided in the "parameter value assignment" section of the PARM3D control file.

*"Layer Files" Section*

The "layer files" section of the PARM3D control file must contain NLAY lines of information. Each of these lines must contain two entries, each of these entries being the name of a file. The first file cited on each line of the "layer files" section should contain a single integer array (the number of rows and columns in which should match that of the model grid). The following name pertains to a file that PARM3D must write. The model layer to which these two files pertain corresponds to the sequence in which they are recorded; that is, the first line of the "layer files" section contains filenames pertaining to layer1, the second line contains filenames pertaining to layer 2, etc.

Integer arrays (which are housed in the first file nominated on each line of the "layer files" section) are used to define zonation within a model grid. Using a set of NLAY integer arrays, every cell within the model domain is assigned an integer value, this value indicating the zone to which the cell belongs. The same zone can appear in different model layers if desired. (Note that zone numbers can be re-assigned in the "parameter value assignment" section of the PARM3D control file as is explained below.)

If the name of an integer array file has an extension other than "*.inu*" then the array is assumed to be stored in ASCII (i.e. text) format. However an "*.inu*" extension indicates to PARM3D that the nominated array is stored in an unformatted (i.e. binary) file. Caution should be exercised in using binary files, however, because different FORTRAN compilers read/write unformatted data using different protocols.

If, for a particular layer, an integer array filename of "none" is supplied, then an array is not read. Instead a dummy zone value of -99999999 is supplied to all cells within the pertinent model layer.

It is PARM3D's job to write a hydraulic property array for each model layer; property values are assigned to cells within each layer using instructions provided in the "parameter value assignment" section of the PARM3D control file. The name of each file to which such an array will be written is supplied as the second entry on each line of the "layer arrays" section of the PARM3D control file. If any extension other than "*.reu*" is supplied for a filename, the array is written in ASCII (i.e. text) format. However if the filename has an extension of "*.reu*" the array is written to a binary file. If a filename of "none" is supplied, then no array is written for that model layer.

If any integer or real array filename cited in the "layer files" section of the PARM3D control file contains a space, it should be enclosed in quotes.

*"Elevation Files" Section*

A PARM3D control file should contain an "elevation files" section only if the ELEVFLAG variable in the "control data" section is provided with a non-zero value.

The "elevation files" section of the PARM3D control file must contain NLAY+1 lines of data. Each such line should contain only one entry, viz. the name of a file containing a real array citing the elevation of the bottom of the pertinent model layer. Layers are arranged from 0 to NLAY. Thus the file cited on the first line should contain cell elevations for the top of layer 1; the file cited on the second line should contain cell elevations for the bottom of layer 1; the file cited on the third line should contain cell elevations for the bottom of layer 2; the file cited on the fourth line should contain cell elevations for the bottom of layer 3, etc. In all cases an extension of "*.reu*" indicates a binary file, while all other extensions indicate an ASCII file; a filename of "none" is not permitted.

*"Parameter Value Assignment" Section*

Each line within the "parameter value assignment" section of the PARM3D control file must contain an instruction through which hydraulic property values are assigned to cells within a zone or layer of the model domain. These instructions are carried out in the order in which they are provided; subsequent instructions can overwrite or modify values assigned to cells through previous instructions.

Instructions fall into a number of broad categories, each of which will now be discussed in detail. (Note that keywords contained within a PARM3D instruction are case-insensitive.)

Default Cell Value

The "parameter value assignment" section must contain one (and only one) line beginning with the word "default". Following that must be a real number. This is the value which PARM3D will assign to all cells within the model domain to which a value is not assigned by any user-provided instruction. It is important to note that, no matter where the "default" instruction appears in the "parameter value assignment" section of the PARM3D control file, the default value assignment operation is carried out only after all other instructions have been implemented.

Layer Value Assignment

All cells within a particular layer of the model grid can be directly assigned a value using an instruction beginning with the word "layer". Some examples are shown below.

```
layer 2 5.643 overwrite
layer 5 layvals.ref geomav
```

**Examples of layer value assignment instructions.**

The first element of a layer value assignment instruction must be the word "layer". Following that must be the layer number to which the instruction pertains; this must

be an integer greater than zero and less than the number of layers in the model. The next entry must be either a real number or the name of a file containing a real array. In the former case the real number is assigned to every cell in the nominated layer; in the latter case cell values are read from the file and assigned to matching rows and columns in the nominated layer of the model grid. (As for the other files discussed above, formatted array storage is assumed unless the extension of the nominated file is "*.reu*".)

If a particular cell in the nominated layer has not been previously assigned a value, the nominated number, or pertinent real array element value, is directly assigned to that cell. However if the cell has been assigned a value through a previous instruction, then five options are available for assimilating the new number into the model property array. The user must supply one of the following keywords (as the final entry of a layer value assignment instruction) in order to exercise the option of his/her choice. These keywords are "overwrite", "geomav", "arithav", "max" or "min".

If the "overwrite" option is provided, the new cell value overwrites any preceding cell value assignments. The "arithav" option specifies that the new cell value will be the arithmetic mean of the previous and new cell values. The "geomav" option specifies that the geometric mean will be taken; note however that PARM3D will report an error condition if the existing or new cell value is zero or negative if this option is selected. For the "max" option, the maximum of the preceding and new value is assigned to the cell, while for the "min" option, the minimum of the preceding and new value is assigned to the cell. Where a cell had not been previously assigned a value the "overwrite" option is used, irrespective of the user-supplied option appearing at the end of the instruction.

Zone Value Assignment

The syntax of a zone value assignment instruction is identical to that of a layer value assignment instruction except for the fact that the keyword "zone" replaces the "layer" keyword. Some examples of zone property assignment instructions are shown below.

```
zone 2 5.643 overwrite
zone 5 layvals.ref geomav
```

**Examples of zone value assignment instructions.**

In the first of the options shown above, the value of 5.643 is assigned to all cells within the model grid which have a zonal value of 2. In the second case any cell possessing a zonal value of 5 is assigned a number equal to that of the cell of same row and column number contained within the real array stored in file *layvals.ref*. As for layer value assignment, new cell values can overwrite or modify existing cell values.

Vertical Property Gradation

Suppose that a band of material (referred to herein as a "hydrogeological unit" or "HGU"), occupying different layers at different locations within the model grid, has hydraulic property values that vary linearly in the vertical direction. PARM3D allows

the user to assign a property array to the top and bottom of that HGU; property values at the centres of cells comprising the HGU itself are then determined by linear vertical interpolation between its top and bottom.

The figure below depicts a HGU which varies in thickness and disposition across part of a model domain.
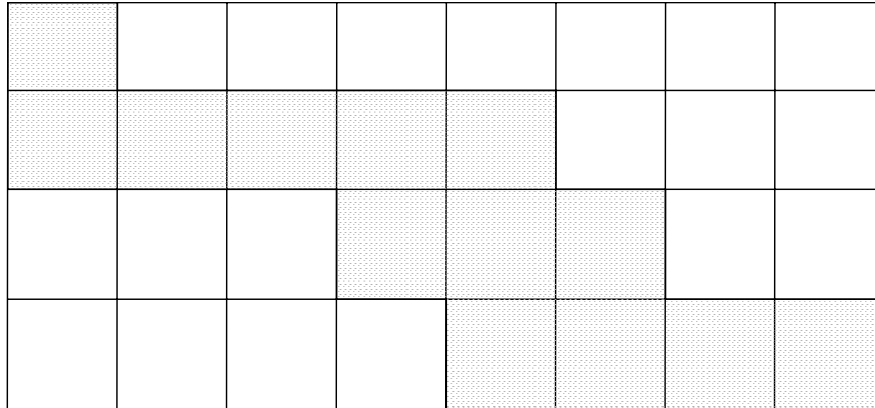


**Vertical section through a model grid showing a hydrogeological unit occupying multiple model layers. (Note that, although depicted as horizontal in the above figure, model layers elevations, as defined in layer elevation arrays, do not need to be constant throughout a model domain.)**

Suppose the shaded material in the above picture is defined by zone *n* of the model domain. In implementing the vertical gradation option for zone *n*, the user must supply the names of two files (or more – see below), each of which contains a real array. These arrays define hydraulic properties at the top and bottom boundaries of the unit (i.e. zone *n*). For any vertical column of the finite difference grid, the hydraulic property at any cell centre lying within the HGU (i.e. any cell centre lying within zone *n*) is then determined through linear interpolation (using vertical distance as the interpolation abscissa) between the top and the bottom of the unit within that column to the pertinent cell centre.

Note the following.

- Linear interpolation can take place on the basis of native or log-transformed property values.

- If, in any particular column, the HGU is only one cell thick, the hydraulic property value assigned to that cell is the arithmetic or geometric mean of the hydraulic property value assigned to the top and bottom of the unit (because the cell centre is half way between the unit's top and bottom under these conditions).

- If a unit is non-contiguous vertically in any column, the top hydraulic property is assigned to the top of the uppermost HGU cell in that column and the bottom hydraulic property is assigned to the bottom of the lowermost HGU cell in that column. Property values assigned to intermediate cell centres are

calculated through linear vertical interpolation between these elevations in the usual manner; intervening cells within the column assigned to other HGUs (i.e. belonging to other zones) are simply ignored in the property assignment process.

More than two property arrays can be supplied as a basis for gradational property assignment. For example three, rather than two, real array files can be provided. In this case the first array is used to assign notional hydraulic properties to the top of the HGU, while the third is used to assign hydraulic properties to the bottom of the HGU. The middle array is used to assign notional hydraulic properties to an elevation equal to the midpoint of the top and bottom of the HGU within any model column. In assigning a property value to a particular cell centre, linear interpolation (on the basis of elevation) then takes place between the pertinent two surfaces which directly overlie and underlie that cell.

More than three property arrays can be provided if desired. Vertical interpolation takes place according to an obvious extension of the principals just outlined.

Examples of vertical property gradation instructions follow.

```
zone 2 grad_arith 2 top2.ref bot2.ref overwrite
zone 3 grad_geom 4 file1.ref  file2.ref  file3.ref  file4.ref arithav
```

**Examples of vertical property gradation instructions.**

Each vertical property gradation instruction must begin with the "zone" keyword, followed by the integer designator of the zone to which the instruction applies. A "grad_arith" or "grad_geom" keyword must follow that. In the former case linear elevation interpolation is undertaken with respect to the native hydraulic property value, while in the latter case interpolation of the log of the hydraulic property value is undertaken (the interpolated value is then back-transformed into native property space before cell assignment).

The next entry in the vertical property gradation instruction must be an integer, specifying the number of real array filenames to follow. The names of the actual files must follow that. As usual, these files are assumed to be formatted unless provided with an extension of ".*reu*".

The final entry on each line must be one of the "overwrite", "geomav", "arithav", "max" or "min" keywords. These have the same meanings as described above for zonal and layer property value assignment.

Vertical Interpolation

This instruction bears some similarity to the vertical gradation instruction; however, unlike the latter, it can apply to both zones and layers. Some examples of this instruction type follow.

```
layer 3 interp_arith arithav
zone 4 interp_geom arithav
```

**Examples of vertical interpolation instructions.**

If the first entry in a vertical interpolation instruction is "layer", then the instruction pertains to the entirety of the nominated layer. Alternatively, if the first entry is "zone", then the instruction applies to all cells within the model grid which belong to the identified zone. The layer or zone number to which the instruction applies is supplied as the second item of the line.

For each cell belonging to the nominated layer or zone, PARM3D searches upwards in the respective grid column for a cell which has already been assigned a value on the basis of a previous instruction. Then it looks downwards within the respective column for a cell which has already been assigned a value. Vertical interpolation then takes place between these already-assigned cells to the new cell. Interpolation is linear. If the third item in a vertical interpolation instruction is "interp_arith", then native property values are vertically interpolated; if it is "interp_geom", the logs of previously assigned values are vertically interpolated (and the resulting interpolated value is then back-transformed prior to cell assignment).

The following should be carefully noted.

- Vertical interpolation takes place using cell elevation as the vertical abscissa. The elevation ascribed to a previously-assigned cell is that pertaining to its midpoint. The elevation ascribed to a new cell for interpolation purposes is also that pertaining to its midpoint.

- If, for a particular cell for which vertical interpolation is required, there is no overlying cell which has been previously assigned a value, the cell is assigned a value equal to that of the nearest underlying cell to which a value has previously been assigned.

- If, for a particular cell for which vertical interpolation is required, there is no underlying cell which has been previously assigned a value, the cell is assigned a value equal to that of the nearest overlying cell to which a value has previously been assigned.

- If there are no overlying or underlying cells within the same grid column which have previously been assigned a value, PARM3D will cease execution with an appropriate error message.

- Underlying or overlying cells from which interpolation takes place do not need to belong to the same zone as the cell to which interpolation takes place.

- Once a value has been calculated through vertical interpolation for a new cell, the actual hydraulic property value assigned to that cell is calculated according to whether the "overwrite", "geomav", "arithav", "max" or "min" keyword is supplied as the last item in the vertical interpolation instruction line.

- If a cell in the nominated layer or zone has already been assigned a value, this fact is ignored when finding overlying/underlying cells from which interpolation must take place to the centre of this cell. However, as mentioned

above, the cell's previously-assigned value is taken into account if the "geomav", "arithav", "max" or "min" keyword is supplied as the last item in the instruction line.

Rezoning

It is often necessary to alter the zone numbers to which cells belong so that subsequent instructions can be used to overwrite previously-assigned cell values. This may be necessary where new hydrogeological structure is introduced to the model grid based on new zone numbers, for example, where a cross-layer HGU is superimposed on previous layers whose properties have been determined through vertical gradation.

Example rezoning instructions are depicted below.

```
layer 1 rezone llay1.inf ignore_zero
layer 3 rezone llay3.inf use_zero
```

**Examples of rezoning instructions.**

As rezoning can only be carried out on layers, the first entry in a rezoning instruction must be "layer"; a layer number must follow that. This must be followed by the name of a file which contains an integer array. As is the normal PARM3D protocol, this will be assumed to be a formatted file unless it possesses an extension of ".*inu*".

Two options are available for assimilation of the new zoning information contained in the nominated integer array. If the final entry in a rezoning instruction is "use_zero", all elements of the new integer array overwrite all elements of the existing zonal structure of the nominated layer. However if the "ignore_zero" keyword is supplied, only non-zero elements of the new integer array are imported; existing zonal values corresponding to zero elements of the new array are thus left unaltered.

There is no limit to the number of rezoning instructions which can be supplied; thus, in the course of parameterising a model domain using PARM3D, rezoning of the entire model grid can take place on multiple occasions if necessary.

*PARM3D Output Files*

PARM3D writes a sequence of real arrays to files nominated in the "layer files" section of its control file. As discussed above, formatted protocol is employed unless a filename extension is ".*reu*" is supplied. If a property file is not required for a certain model layer, PARM3D can be informed of this by supplying the pertinent filename as "none".

**Uses of PARM3D**

PARM3D can be used to construct a complex parameterisation of a model domain where this parameterisation is based on assemblages of real arrays. Hydraulic properties can be assigned to real arrays using pilot points or zones. This can be useful when calibrating such a model using PEST. In this case the batch or script file

run by PEST as "the model" will be comprised of one or more instances of PARM3D followed by MODFLOW and/or MT3D. One or more instances of FAC2REAL or INT2REAL would be run ahead of PARM3D to build hydraulic property arrays on the basis of pilot points or zones, before incorporating these into the model.

If calibration of such a model were undertaken using regularised inversion, regularisation constraints could be added using the GENREG utility.

**See Also**

See also FAC2REAL, INT2REAL, GENREG, PESTPREP and PPKREG.

# PESTPREP

**Function of PESTPREP**

Program PESTPREP undertakes the laborious task of preparing a PEST control file and a PEST instruction file for a MODFLOW or MT3D run. The laborious nature of this work arises from the fact that, for a long transient run, it normally involves the processing of a great deal of observation data. However if PESTPREP is used the entire process is automated. On the assumption that the "model" to be calibrated consists of MODFLOW or MT3D followed by program MOD2OBS, or of MODFLOW followed by BUD2SMP followed by SMP2SMP, PESTPREP prepares an instruction file to read the MOD2OBS or SMP2SMP output file, and builds a PEST control file containing measured values as listed in the "measurement" bore sample file on which the model-generated bore sample file produced by MOD2OBS or SMP2SMP is based; parameters listed in a set of template files are also recorded in the PESTPREP-prepared PEST control file.

Note that the use of PESTPREP is not just restricted to the use of PEST with MODFLOW and MT3D. Together with FEM2SMP and SMP2SMP (and maybe PPK2FACF and FAC2FEM for pilot-point-based parameterisation) it can also expedite the use of PEST with MicroFEM.

**Using PESTPREP**

PESTPREP's use as a PEST preprocessor is predicated on the assumption that the model run by PEST produces as one of its output files a bore sample file in which model outputs are spatially and temporally interpolated to measurement dates and times. Recall that MOD2OBS writes a bore sample file containing model-generated heads (or drawdowns, etc) interpolated to the same sites and times at which field measurements were made, the latter being supplied in a "measurement" bore sample file. SMP2SMP, when run following BUD2SMP, lists model inflows/outflows interpolated to the measurement dates and times of these same quantities. SMP2SMP when run following FEM2SMP performs a similar task in the MicroFEM context. It is the role of PESTPREP to write an instruction set by which the MOD2OBS or SMP2SMP-generated bore sample file can be read, and to generated a PEST control file whose corresponding observation values are extracted from the "measurement" bore sample file.

PESTPREP begins execution with the prompts:-

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

The first is the bore sample file containing measured data. The second is the bore sample file generated by MOD2OBS or SMP2SMP as part of the model. If a filename file [files.fig] is present in the current directory, the first of the above prompts is accompanied by a default bore sample filename which can be accepted through simply pressing the <Enter> key. Note that it is essential that the above bore sample

files be "paired" in the sense that the latter is generated by MOD2OBS or SMP2SMP on the basis of the former as part of the composite model run.

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

```
 Use numbers or bore identifiers for observation names?  [n/b]:
```

If "n" is selected, observations are named from 1 to 99999999 in order of their appearance in the MOD2OBS or SMP2SMP-generated bore sample file (which will also be the order of their appearance in the measurement bore sample file upon which the MOD2OBS or SMP2SMP-generated bore sample file is based). Alternatively, select "b" for greater ease in relating observation names to actual measurements. In that case PESTPREP prompts:-

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

where *n* is a number from 3 to 9. If "f" is typed in response to the above prompt, PESTPREP generates observation names by taking the first *n* characters of the bore identifier and affixing the suffix "*_mm*" to its name, where *mm* signifies the *mm*'th sample pertaining to that bore as read from the MOD2OBS or SMP2SMP-generated bore sample file. PESTPREP determines *n* in the above prompt through counting the maximum number of observations pertaining to any bore and thus determining how many of the twelve characters available in an observation name can be assigned to sample numbering in this fashion. If this method of assigning observation names does not result in a unique set of names due to the fact that different bore identifiers have the same first *n* letters in common, PESTPREP informs the user of this. He/she is then prompted for an alternative method of observation name generation.

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

PESTPREP's next prompt is:-

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

Once it is supplied with this name (preferably with an extension of ".ins") PESTPREP generates the instruction set by which the MOD2OBS or SMP2SMP-generated bore sample file can be read. Then it gathers the names of the various parameters involved in the current parameter estimation problem by reading all template files involved in the current PEST run. It prompts:-

```
 How many template files are there?
 Enter name for template file # 1:
 Enter name of corresponding input file:
```

```
 Enter name for template file # 2:
 Enter name of corresponding input file:
   etc.
```

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

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

and finally for the command which must be used in the "model command line" section of this file:-

```
 Enter command which PEST will use to run model:
```

It then writes the PEST control file.

PESTPREP observes a number of conventions with respect to the assigning of parameter groups, initial values and parameter value bounds when writing the PEST control file. These conventions are as follows:-

- If the name of a parameter begins with a letter of the alphabet it is assigned to a parameter group whose name is that letter. Thus parameters *prop1*, *prop2* and *prop3* are assigned to a parameter group whose name is *p*. However if a parameter name begins with any other character, it is assigned to a group named *other*.

- Parameters whose names begin with *v* (for *vcont*), *t* (for *transmissivity*), *h* (for *hydraulic conductivity*), *p* (for *permeability*) or *k* (for *konductivity*) are logarithmically transformed. They are assigned an initial value of 1.0, a lower bound of $10^{-10}$, and an upper bound of $10^{10}$.

- Parameters whose names begin with *s* (for *storage coefficient* or *specific yield*) are also logarithmically transformed. They are assigned an initial value of 0.1, a lower bound of $10^{-10}$, and an upper bound of 0.3

- All other parameters are assigned an initial value of 1.0, a lower bound of $-10^{10}$, and an upper bound or $10^{10}$.

Note also that, on the assumption that *r* stands for *recharge*, the derivative increment for members of parameter group *r* is calculated using the *rel_to_max* method.

**The user will probably wish to alter many of these settings once the PEST control file has been created.**

PESTPREP assumes that only one instruction file is required by the inversion process, this being the one that it has written itself to read the MOD2OBS or SMP2SMP-generated bore sample file, of which it also knows the name.

Once PESTCHEK has written the PEST control file, the status of the PEST input dataset can be immediately checked using program PESTCHEK. Alterations can be made to that file using a text editor.

Note that when using PESTPREP, or any other member of the Groundwater Data Utilities Suite, the user can "backtrack" in execution by responding to any prompt by simply typing "E" or "e", followed by <Enter>; "e" stands for "escape".

**Uses of PESTPREP**

PESTPREP automates most of the laborious work required in the preparation of a PEST run used for the calibration of a MODFLOW, MT3D, MicroFEM, or other model. Because it can rapidly process the large amounts of data that often accompany transient model calibration, it can accomplish in seconds that which would take hours to accomplish in any other way.

**See Also**

See also FEM2SMP, MOD2OBS, SMP2DAT and SMP2SMP.

# PMP2INFO

**Function of PMP2INFO**

PMP2INFO tabulates the amount of water pumped between two user-specified times from a list of user-specified bores. It obtains the information which it needs to calculate pumped volumes by reading a bore pumping file. Where the beginning or end of the user-supplied time interval does not coincide with sampling times recorded in the bore pumping file, PMP2INFO linearly interpolates data contained in this file to the user-specified time-interval endpoints. It writes its output data (including bore coordinates) to a bore information file.

**Using PMP2INFO**

Program PMP2INFO will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities.

Upon commencement of execution PMP2INFO prompts:

```
 Enter name of bore coordinates file:
```

Enter the appropriate filename. If PMP2INFO located a filename file (`files.fig`) within the current directory, a default bore coordinates filename may appear in the above prompt. In this case you should either press <Enter> to accept the default, or supply another filename yourself. Note that there is no need for the bore coordinates file read by PMP2INFO to include the optional layer number column; if this column is present, it is ignored.

PMP2INFO then asks:

```
 Enter name of bore listing file:
```

The bore listing file should contain that subset of the bores cited in the bore coordinates file for which pumping figures are required. If pumping figures are required for all bores cited in the bore coordinates file, the latter can be resubmitted as a bore listing file; PMP2INFO only reads the first column of a bore listing file.

Next PMP2INFO prompts for the name of the bore pumping file which holds the data from which it must calculate individual pumping volume figures. See Section 2.6 of Part A of this manual for the specifications of a bore pumping file. PMP2INFO prompts:

```
 Enter name of bore pumping file:
```

Supply the appropriate filename. If a filename file holding the name of a bore pumping file is present in the current directory, PMP2INFO includes the default

filename obtained from this file in the above prompt. In this case you should either press <Enter> to accept the default or type in the correct filename.

The time interval over which pumped volumes for each bore are to be calculated must next be supplied. The PMP2INFO prompts are:

```
Enter time interval starting date  [dd/mm/yyyy]:
Enter time interval starting time  [hh:mm:ss]:
Enter time interval finishing date [dd/mm/yyyy]:
Enter time interval finishing time [hh:mm:ss]:
```

(Note that the date representation format depends on the contents of the settings file `settings.fig`.) If the date and time corresponding to the beginning or end of the time interval does not coincide with a sampling date and time as recorded in the bore pumping file, PMP2INFO carries out a linear interpolation of data contained in the latter file to the user-specified interval beginning or end point. The user-specified time interval can be of any duration; however precision may be lost if the length of the interval is very small relative to the time between readings. Note also that if the start of the user-specified time interval precedes the first pumpage reading for a particular bore and/or the end of the time interval postdates the last pumpage figure recorded for a particular bore in the bore pumping file, PMP2INFO will be unable to calculate the amount of water pumped from that bore during the interval.

PMP2INFO writes its calculated pumped volumes to a bore information file whose name must be specified in response to the prompt:

```
Enter name for output bore information file:
```

Then PMP2INFO prompts:

```
Record any uninterpolated bores to output file?  [y/n]:
```

If you respond to this prompt with "n", PMP2INFO will omit from the bore information file which it generates any bores for which it cannot calculate the required extraction volume during the user-supplied time interval. Otherwise it will include these bores in its output file, denoting its inability to perform the required calculation by one of a set of codes replacing the calculated pumped volume figures. Thus if the beginning of the user-supplied pumping interval precedes the first pumpage reading for a bore as recorded in the bore pumping file, PMP2INFO writes the text "`before_first_sample`" in place of the pumped volume for that bore. If the end of the user-defined pumpage interval postdates the last pumping reading for a particular bore, the extracted volume for that bore is recorded as "`after_last_sample`". If a bore cited in the bore listing file was not found in the bore pumping file, its extracted volume is recorded as "`not_in_pumping_file`".

The following figure shows part of a bore information file generated by PMP2INFO. Note that borehole coordinates (read from the bore coordinates file) comprise the second and third columns of a PMP2INFO-generated bore information file.

```
40236        426276.0160    7256191.549    0.00000
40240        423156.2920    7256758.984    after_last_sample
40241        423242.3010    7256451.838    after_last_sample
40402        430721.1320    7254368.971    0.00000
40403        430497.9760    7254091.023     61.935
40407        431574.6490    7252097.156     37.793
40409        430970.8520    7254954.668     26.196
40413        430467.7110    7254521.561     2.6957
40414        430299.8120    7254428.439     16.998
40416        431617.7600    7254711.861     194.06
40418        431028.5810    7254647.337    after_last_sample
40423        429652.7360    7254701.978     7.1839
40429        431812.9540    7254989.651    0.00000
40431        430664.1160    7254553.345     38.693
40433        430718.1960    7254953.401     3.9957
40435        430800.9390    7255261.436     114.43
40437        429679.8910    7254886.556     4.0675
40440        433024.5250    7254072.732     43.672
40806        423429.9890    7252977.065    0.00000
40810        423390.7910    7254976.133     9.6186
40841        422804.9000    7254326.931    0.00000
40842        423279.0220    7254883.235    0.97154
40844        423878.6440    7253071.716    0.00000
```

**Extract from a bore information file generated by PMP2INFO.**

**Uses of PMP2INFO**

PMP2INFO can assist in the undertaking of simple water balance studies over various parts of an aquifer. For example calculations of pumped volumes over dry periods where aquifer recharge is expected to be zero, supplemented by measured water level falls in observation bores over the same period (together with some simple assumptions concerning aquifer inflow and outflow) can lead to an estimate of aquifer specific yield. Another non-modelling application, made possible by the fact that the bore information file generated by PMP2INFO provides bore coordinates in its second and third columns, is the posting of pumping information on a map of the study area, perhaps with symbol size proportional to extracted volume.

In the modelling context, the bore information file generated by PMP2INFO can be used by program PT2ARRAY to assign pumping rates to model cells containing bores. Thus the joint use of PMP2INFO and PT2ARRAY can greatly facilitate the time-consuming and laborious task of incorporating borehole pumping data into a groundwater model.

**See Also**

See also PMPCHEK, PT2ARRAY.

# PMPCHEK

**Function of PMPCHEK**

PMPCHEK checks the integrity of a bore pumping file, reading the file in its entirety and writing any errors it finds to the screen.

**Using PMPCHEK**

Program PMPCHEK will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities.

PMPCHEK begins execution with the prompt:

```
 Enter name of bore pumping file:
```

to which you should respond with the appropriate filename. If a filename file (`files.fig`) holding the name of a bore pumping file is present in the current directory, PMPCHEK includes the default filename obtained from this file in the above prompt. In this case you should either press <Enter> to accept the default or type in the correct filename.

PMPCHEK then reads the nominated file, writing any errors that it finds to the screen (up to a maximum of 40 errors). It checks that the following conditions are met in the bore pumping file:

- That every line contains sufficient items, and that each date, time and pumping figure can be read correctly.

- That all dates and times are valid.

- That data for all bores are sequential, and that the beginning of one time interval coincides with the end of the previous interval for subsequent entries citing the same bore.

- That all bore identifiers are of 10 characters or less in length.

**Uses of PMPCHEK**

Those Groundwater Data Utility programs which obtain part of their data requirements from a bore pumping file carry out rudimentary error checking as they read the file; however their error checking is not exhaustive. Furthermore, on encountering an error condition they usually write the error (and line number) to the screen and then cease execution. Thus if a bore pumping file has more than one error the next error will go undetected until the first error is rectified and the utility is run again. The detection (and rectification) of errors in this "one-by-one" fashion can

become time-consuming and frustrating if a bore pumping file has more than just a few errors.

PMPCHEK was written to record and report all errors at once. When writing errors to the screen it reports the file line number on which each error condition occurs so that a user may easily locate and rectify these errors. Note that a user can redirect PMPCHEK screen output to a file (using the ">" symbol) for a more permanent record of the errors.

It is strongly recommended that PMPCHEK be used to establish the integrity of a bore pumping file before the latter is used by any of the Groundwater Data Utilities.

**See Also**

See also MKPMP1, PMP2INFO.

# PPCOV

**Function of PPCOV**

PPCOV reads a pilot points file and a geostatistical structure file. On the assumption that each pilot point represents a model parameter, it writes a covariance file for these parameters based on geostatistical structures pertaining to various sets of these points. Points within the pilot points file can belong to one such set, or many different sets, each potentially being characterised by a different structure.

**Using PPCOV**

PPCOV commences execution by prompting for the name of a pilot points file.

```
 Enter name of pilot points file:
```

If the name of such a file is cited in a *files.fig* file contained within the current working directory, this filename will appear with the above prompt; it can then be accepted by simply pressing the <Enter> key.

The format of a pilot points file is described in part A of this manual. As noted there, the first column of a pilot points file contains pilot point names (10 characters or less in length). The next two columns contain pilot point eastings and northings. Following these are zone numbers, followed by the values assigned to pilot points. PPCOV ignores data provided in the last column of this file. However it uses zone numbers to subdivide pilot points into groups. Pilot points within each group are assumed to be characterised by a single geostatistical structure. There is assumed to be *no statistical dependence* between groups; thus covariances between pilot point parameters from different groups are assumed to be zero.

The names of parameters pertaining to pilot points are assumed to be either the same as the names of the pilot points themselves, or to be derived from pilot point names by addition of a prefix. See below.

PPCOV next prompts:-

```
 Enter minimum allowable separation for points in same zone:
```

This allows the user to check that no two pilot points within the same zone are closer than he/she thinks they are, or even inadvertently superimposed. If any two points are closer than the minimum separation supplied in response to the above prompt, PPCOV will list the offending points and then cease execution. In most cases the appropriate response to the above prompt is zero.

PPCOV's next prompt is:-

```
 Enter name of structure file:
```

The format of a structure file is presented in part A of this manual. It contains specifications for one or a number of geostatistical structures, each of which can comprise a nugget and one or more nested variograms (which may be anisotropic if desired). Each structure has a name; one such structure is assigned to each zone

identified in the pilot points file in response to the following sequence of prompts issued by PPCOV:-

```
Enter structure to use for pilot point zone 5:
Enter structure to use for pilot point zone 6:
etc
```

Zone numbers are as supplied in the pilot points file, and are listed in increasing order in the above series of prompts. In response to each of these prompts, the user should supply the name of a geostatistical structure whose specifications are provided in the structure file.

PPCOV's final two prompts are:-

```
Enter name for output matrix file:
Enter pilot point prefix for parameter name (<Enter> if none):
```

The name of a file to which the covariance matrix will be written is supplied in response to the first of the above prompts. The prefix by which pilot point names are converted to parameter names is supplied in response to the second of the above prompts.

The format of the file to which the covariance matrix is written is the same as that used by PEST matrix manipulation utilities. See either the PEST manual or the addendum to the PEST manual for details.

**Uses of PPCOV**

PPCOV can be used for building a C($\mathbf{p}$) (i.e. a parameter covariance) matrix for the use of PEST utilities such as RESPROC, PREDERR, PARAMERR, PREDVAR1 and PREDVAR2 in quantifying the error variance of model predictions. As such it furnishes the basis for calculation of the contribution to predictive error variance made by the inability of the model calibration process to capture all system hydraulic property detail. In most modelling contexts this is the dominant contributor to model predictive error.

**See Also**

See also PPK2FAC, FIELDGEN and GENREG.

# PPK2FAC

**Function of PPK2FAC**

PPK2FAC generates a set of kriging factors for use in spatial interpolation from a set of pilot points to a MODFLOW/MT3D finite-difference grid. Kriging factors are based on user-supplied, nested variograms, each with an arbitrary magnitude and direction of anisotropy. Different variograms can be used for spatial interpolation in different parts of the model domain. PPK2FAC also writes a MODFLOW-compatible real array depicting kriging standard deviations over the model domain, as well as a "regularisation information file" which can be used to introduce geostatistically-based regularisation constraints to a parameter estimation problem.

Generation of MODFLOW and MT3D input arrays based on PPK2FAC-generated kriging factors is carried out by other programs of the Groundwater Data Utilities such as FAC2REAL and FAC2MF2K. Separation of the time-consuming, factor-generation process from the array construction process facilitates automatic parameter estimation based on pilot points using software such as PEST, for kriging factors are unchanged as values assigned to the pilot points are adjusted through the parameter estimation process.

Regularisation information recorded by PPK2FAC is used by program PPKREG; PPKREG modifies an existing PEST control file containing pilot-point-based parameters, adding regularisation constraints based on the geostatistical structure of the area, and the distances between the pilot points on which parameterisation of the model domain is based.

**Running PPK2FAC**

*Structure File*

Before the operational details of PPK2FAC are presented, a "structure file" will be described. PPK2FAC reads such a file in order to ascertain the geostatistical characteristics of the areas in which spatial interpolation is to be carried out. Use of a geostatistical structure to characterise the spatial variation of a hydraulic property assumes that values taken by that property are spatially correlated, and that the degree of correlation between values at two different points is dependent only on their separation. Furthermore, it is assumed that this inter-point distance-dependence can be described by one or more nested variograms; an optional uncorrelated component (a "nugget") of the hydraulic property field can also be represented.

A structure file is depicted below.

```
STRUCTURE struct1
  NUGGET 0.0
  TRANSFORM log
  NUMVARIOGRAM  2
  VARIOGRAM var1 0.6
  VARIOGRAM var2 0.3
END STRUCTURE

VARIOGRAM var1
  VARTYPE 2
  BEARING 72
  A 3000
  ANISOTROPY 13.5
END VARIOGRAM

VARIOGRAM var2
  VARTYPE 1
  BEARING 72
  A 4000
  ANISOTROPY 5.0
END VARIOGRAM


STRUCTURE struct2
  NUGGET 0.0
  TRANSFORM none
  MEAN 23.5
  NUMVARIOGRAM  1
  MAXPOWERVAR 10000
  VARIOGRAM var3 .005
END STRUCTURE


VARIOGRAM var3
  VARTYPE 4
  BEARING 20
  A 0.005
  ANISOTROPY 1.0
```

**A structure file**

A structure file is subdivided into different segments. Segments are of two types - "structure segments" and "variogram segments". Each of these segments must be assigned a unique name. This name is written to the first line of the segment following the word STRUCTURE or VARIOGRAM; it must be 10 characters or less in length. A structure segment must end with the words END STRUCTURE; a variogram segment must end with the words END VARIOGRAM. Note that although some words in the above example are shown capitalised, every item within a structure file is, in fact, case insensitive.

Within each structure or variogram segment, data is supplied through the use of keywords (capitalised in the above example). Wherever a keyword is supplied it must be followed by the value of the variable which the keyword represents. For all keywords except VARIOGRAM only one entry is required following the keyword itself; however where the keyword VARIOGRAM occurs within a STRUCTURE segment, two variables must follow.

Within any segment of a structure file, keywords and corresponding values can be supplied in any order. However there is one exception to this rule; this is that the NUMVARIOGRAM keyword must precede the VARIOGRAM keyword in a STRUCTURE segment.

Within each segment of a structure file, each keyword should be cited only once. However there is also one exception to this rule; this is that the VARIOGRAM keyword within a STRUCTURE segment must be repeated NUMVARIOGRAM times.

Some keywords are mandatory; others can be omitted where they are not required.

Keywords that can appear in a STRUCTURE segment are NUGGET, MEAN, NUMVARIOGRAM, TRANSFORM, VARIOGRAM and MAXPOWERVAR. NUGGET, MEAN and MAXPOWERVAR are optional.

Each STRUCTURE segment specifies the geostatistical components of the random field which characterises the spatial distribution of some hydraulic property over all or part of the model domain. These components are a NUGGET (optional) and up to five VARIOGRAMs (the number of variograms contributing to any structure being provided by the variable NUMVARIOGRAM). The value supplied for the NUGGET specifies the contribution made to the total, nested variogram by a random field lacking any spatial correlation. The contribution made to the total, nested variogram by each of the NUMVARIOGRAM variograms cited in the STRUCTURE segment is provided after each VARIOGRAM keyword - see below.

If the variograms and nugget comprising the geostatistical structure pertain to the native value of a hydraulic property, then TRANSFORM should be set to "none". However if they pertain to the log (to base 10) of a hydraulic property, then TRANSFORM must be provided as "log". (Note that the use of log variograms, though conceptually correct when describing the spatial variation of hydraulic properties such as transmissivity and hydraulic conductivity, presents certain problems when working with MODFLOW-2000. Fortunately, as is described elsewhere, these problems are easily overcome. See Section 5 of Part A of this manual for further details.)

If "simple kriging" is to be carried out, then a mean value for the hydraulic property represented by the geostatistical structure must be provided in each STRUCTURE segment following the keyword MEAN. However if "ordinary kriging" is to be carried out, a mean hydraulic property value does not need to be provided; hence the MEAN keyword, if present, is ignored. In general, it is better to undertake ordinary kriging than simple kriging; see Section 5 of Part A of this manual for a further discussion. *It is important to note that if TRANSFORM is set to "log", then the mean property value supplied after the MEAN keyword must pertain to the log-transformed property distribution.*

Two items of information must follow each incidence of the VARIOGRAM keyword within a STRUCTURE segment. The first is the name of a variogram. This must be 10 characters or less in length; specifications for the named variogram must be supplied elsewhere in the file within a VARIOGRAM segment. The second item of

information following each VARIOGRAM keyword within a structure segment is the contribution made to the overall structure by that variogram. The sum of the contributions made by all variograms, plus the nugget, is equal to the sill of the nested variogram comprising the structure, ie. its long-range asymptote. It is important to remember that when TRANSFORM is set to "log", variogram contributions must pertain to the log of the hydraulic property field.

MAXPOWERVAR is an optional variable. It is only used if one of more of the variograms cited in a STRUCTURE segment are "power" variograms. This type of variogram does not have a sill. The sill of a variogram, or of a set of nested variograms comprising a structure, is equal to the covariance of the regionalised variable represented by the structure. Thus any structure that possesses a power variogram component does not have a finite covariance. This brings with it certain numerical difficulties when kriging factors are computed based on that structure. Fortunately, these difficulties are easily overcome by assuming a suitably high covariance; unless a value for MAXPOWERVAR is supplied by the user, PPK2FAC assumes a value of 10000 for the structure covariance. Under most circumstances, the assumed covariance makes no difference to calculated kriging factors. However if the parameters of a power variogram are such that variogram values become this high at distances within the dimensions of the study area, then a higher covariance should be assigned to the MAXPOWERVAR variable.

Keywords that can appear within a VARIOGRAM structure are VARTYPE, BEARING, A and ANISOTROPY. All of these are mandatory.

VARTYPE specifies the type of variogram; this must be supplied as either "1", "2", "3" or "4" indicating a spherical, exponential, Gaussian or power variogram respectively. See Section 5 of Part A of this manual for the definition of each of these variogram types. Note that, for reasons outlined in that section, use of a Gaussian variogram is not recommended; also, care should be taken when using a power variogram. The "A" keyword pertains to the *a* variable appearing in each of equations 5.1 to 5.4 of Part A of this manual; for all but the power variogram this is related to the range of the variogram.

BEARING specifies the angle (in degrees) between north and the axis of anisotropy (normally the direction of greatest uniformity) of the random field characterised by the variogram. ANISOTROPY specifies the ratio of the range in this direction to the range in a direction at 90 degrees to BEARING. If BEARING does, indeed, indicate the direction of elongation of the ellipse of anisotropy, then ANISOTROPY will be greater than 1. However, if desired, the user can enter BEARING as the direction of the short axis of the ellipse of anisotropy; in this case ANISOTROPY will be smaller than 1. Under isotropic conditions, ANISOTROPY should be set to 1.

Note that if the name of a variogram is cited in a STRUCTURE segment, then specifications for that variogram must be provided in a VARIOGRAM segment. However if specifications for a variogram are supplied in a VARIOGRAM segment, and that variogram is cited nowhere within a STRUCTURE segment, PPK2FAC will not complain.

*Using PPK2FAC*

PPK2FAC execution is initiated by typing its name at the screen prompt. However if a "settings file" is not present in the directory from which it is run, and/or if no specification is set within that file for the `colrow` variable, PPK2FAC will cease execution immediately with an appropriate error message.

When replying to PPK2FAC's prompts in the manner discussed below, do not forget that, as with all programs of the Groundwater Data Utilities, you can "backtrack" to the previous prompt by pressing the "e" key followed by <Enter> in response to the current prompt.

PPK2FAC first asks for the name of the grid specification file pertaining to a MODFLOW model.

```
 Enter name of grid specification file:
```

See Section 2 of part A of this manual for the details of this file. If the name of a grid specification file is cited in a filename file (named `files.fig`) residing in the directory from which PPK2FAC was run, then that name will be included in the above prompt; then you need only press the <Enter> key for this name to be accepted.

Next PPK2FAC prompts:-

```
 Enter name of pilot points file:
```

A pilot points file is described in Section 2 of Part A of this manual. If the name of a pilot points file is cited in a filename file (named `files.fig`) residing in the directory from which PPK2FAC was run, then that name will be included in the above prompt; simply press the <Enter> key for this filename to be accepted.

PPK2FAC then prompts:-

```
 Enter minimum allowable points separation:
```

Enter a value of 0.0 or greater. PPK2FAC calculates the distances between all pairs of points cited in the pilot points file that belong to the same zone. If any of these distances are less than or equal to the distance entered in response to the above prompt, PPK2FAC will list the names of the pertinent points to the screen. When all such pairs of points have been listed, PPK2FAC will then terminate execution with an appropriate message. As was discussed in Section 5 of Part A of this manual, if pilot points are too close together, problems can be encountered in calculating kriging factors, especially if the Gaussian variogram is employed.

PPK2FAC next asks for the name of a MODFLOW-compatible integer array file:-

```
 Enter name of zonal integer array file:-
```

As is usual for programs of the Groundwater Data Utilities, if an integer array filename has an extension of "inf" then it is assumed to be formatted; if it has an extension of "inu" it is assumed to be unformatted. If any other extension is supplied,

PPK2FAC will ask whether the file is formatted or unformatted. Note also that if the file is formatted then, depending on the setting of the `colrow` variable in file `settings.fig`, PPK2FAC may expect a dimensional header in the file preceding the integer array.

The zonal integer array defines zones within a model domain. Hence, depending on the application, the zonal integer array may be uniform, or it may contain a number of different integers, each representing a different geological unit occurring within the study area.

Next PPK2FAC prompts for the name of a structure file:-

```
 Enter name of structure file:
```

The specifications of a structure file were provided in the previous section. It contains the information required to specify the geostatistical structure pertaining to one or a number of hydraulic properties in one or a number of geological units occurring within the model area; the disposition of these units is defined by the elements of the zonal integer array.

PPK2FAC then makes a list of all integers cited within the zonal integer array. For each one of these zone-defining integers, it asks the following series of questions:-

```
 For zone characterised by integer value of n:-
 Enter structure name (blank if no interpolation for this zone):
```

Type in the name of a geostatistical structure supplied in the structure file. This will be the structure upon which kriging factors will be based for model cells lying within this zone. For each such cell a number of factors is calculated, linking that cell to some or all of the pilot points assigned to that zone. When the actual spatial interpolation is later carried out (for example by program FAC2REAL), these factors will be multiplied by the values pertaining to respective pilot points and added together to form the interpolated value at the centre of the cell. Note that if you wish that factors not be calculated for cells within this zone, simply press the <Enter> key in response to the above prompt.

The next question asked by PPK2FAC pertaining to the current zone is:-

```
 Perform simple of ordinary kriging [s/o]:
```

In general, for the reasons explained in Section 5 of Part A of this manual, it is better to undertake ordinary kriging. Note, however, that if you wish to calculate kriging factors for simple kriging, then a mean value for the hydraulic property over the zone must have been supplied in the pertinent structure in the structure file.

The next zone-specific question is:-

```
 Enter search radius:-
```

Enter a positive number; when looking for pilot points for which to calculate kriging factors for each cell, PPK2FAC will restrict its search to a distance from each cell

centre equal to the search radius supplied here. In general, it is best that this search radius be large enough to ensure that "quite a few" pilot points are linked to each cell centre. If you wish that all pilot points assigned to a particular zone be linked to each cell centre within that zone (which is often the case), then enter a number as large as you like in response to the above prompt (any number larger than the largest dimension of the zone would be suitable). In general, it is best to err on the side of caution and to make the search radius large enough to include many, rather than few, pilot points, for this ensures a smooth interpolated property field. However if too many points are used, calculation of the kriging factors can be very slow. Also, it may be desirable in some instances to restrict the number of points used for interpolation to any one cell centre in order to allow a little more spatial property variation within the zone.

PPK2FAC's next two prompts also pertain to the number of pilot points used in calculating interpolated property values at cell centres.

```
Enter minimum number of pilot points to use for interpolation:
Enter maximum number of pilot points to use for interpolation:
```

The answer to the first of the above questions must be at least 1. If ever PPK2FAC fails to find the specified minimum number of pilot points within a distance of one search radius from any cell centre, no kriging factors will be calculated for that cell centre; the "interpolated" hydraulic property for that cell will then be assigned a dummy value by program FAC2REAL when it applies the kriging factors to actually carry out the interpolation.

By supplying an appropriate answer to the second of the above requests, you can restrict the number of points used to perform spatial interpolation to any one cell (ie. only the closest *n* points will be used, where *n* is supplied in response to the prompt). As stated above, this may sometimes be desirable in order to reduce the time required to calculate kriging factors (this time rises rapidly if more than about 25 points are used), and to allow a little more spatial variation of the interpolated property field.

In answering the above three questions, the "safest" option is to supply a very large value for the search radius (far larger than the largest dimension of the model domain), a value of 1 in answer to the second of the above prompts, and use the third answer to limit the number of pilot points used for interpolation, if you wish. However, unless there are more than about 10 pilot points within a zone, there is really nothing to be gained by limiting the number of points used in spatial interpolation. In that case, simply answer the third of the above prompts by supplying a number that is larger than the number of points listed in the pilot points file; any number up to 500 will do.

Before calculating kriging factors, PPK2FAC prompts the user for its output filenames. The first of these prompts is:-

```
 Enter name for interpolation factor file:
 Is this a formatted or unformatted file? [f/u]:
```

Kriging factors are written to a special file named an "interpolation factor file"; this file can be read by programs FAC2REAL and FAC2MF2K documented herein. The

file can be written as a text (ie. "formatted") or binary (ie. "unformatted") file. The latter method may prove useful where FAC2REAL is used in conjunction with MODFLOW and/or MT3D to undertake pilot-point-based parameter estimation under the control of PEST. When this is done, the "composite model", as run by PEST, must be executed many times during the overall parameter estimation process. On each occasion that this composite model is run, FAC2REAL will need to re-read the interpolation factor file. If this is a binary file, rather than a text file, the time required to read this file can be significantly reduced, especially if the model grid is large.

Next PPK2FAC prompts:-

```
 Enter name for output standard deviation array file:
```

A MODFLOW-compatible real array will be written to this file. As is explained in Section 2 of Part A of this manual, this can be a formatted or unformatted file. If the extension is "ref" PPK2FAC (like other programs of the Groundwater Data Utilities) assumes that the file is formatted; however if a filename with an extension of "reu" is supplied, then PPK2FAC assumes that the file is a binary file. If neither of these extensions is supplied, PPK2FAK prompts the user for the nature of the file.

The standard deviation array file contains the square root of the "kriging variance" calculated for every cell centre for which kriging factors are calculated. Because it contains a MODFLOW-compatible real array, this file should be capable of importation into most MODFLOW GUI's for visualisation and display.

Finally PPK2FAC prompts:-

```
 Enter name for regularisation information file:
```

This file is written for the benefit of program PPKREG which adds regularisation data to a PEST control file built to estimate parameters based on the same pilot points as are listed in the pilot points file read by PPK2FAC. The regularisation information file lists the names of all of the pilot points cited in the pilot points file. Following this is a matrix in which, for various pairs of pilot points, a variogram value is calculated. Note the following:-

1. A variogram value is calculated for a pair of points only if both points lie within the same zone.

2. The geostatistical structure pertaining to that zone (which may include a number of nested variograms and a nugget) is used to calculate the variogram value pertaining to the pair of points. The distance used in this calculation is the distance between the respective pilot points.

3. A variogram value is calculated for a particular pair of points only if there is at least one cell within the model grid for which a kriging factor has been calculated linking both of those points to that cell. If two particular pilot points are "too far apart" (as defined by the search radius or by the "maximum number of points for interpolation" criterion), then a variogram value for this pair of points is not calculated by PPK2FAC. Hence a

relationship between this pair of points is not included as a regularisation prior information item in the PEST control file generated by program PPKREG. In some instances, this will serve to reduce the number of such prior information items to a manageable level.

As soon as a response is provided to the last of the above prompts, PPK2FAC commences its calculation of kriging factors. For each zone for which these factors are required, PPK2FAC displays a screen message similar to this:-

```
Number of pilot points for this zone    =     8
 Mean data value for these pilot points  =   12.456
 Data standard deviation for these points =   3.4563
 Working....
 No. of grid points to which factors were calculated =  2710
```

Note that, as is explained in Section 5 of part A of this manual, kriging factors are independent of the hydraulic properties assigned to pilot points. Hence the property values associated with pilot points, as read from the pilot points file, are not used in the calculation of kriging factors; they are only used in calculating the above statistics. Thus the same pilot points file used by program PPK2FAC for the calculation of kriging factors, does not need to be supplied to program FAC2REAL for carrying out the actual spatial interpolation, or to program FAC2MF2K for the modification of an existing MODFLOW-2000 dataset by inclusion of pilot-point-based parameters. However the pilot points file supplied to either of these programs must cite the same pilot points as those read by PPK2FAC, and must list them in the same order. If this is not done, the respective program will cease execution with an appropriate error message.

PPK2FAC carries out the first step of a spatial interpolation process based on kriging. The second step is carried out by either FAC2REAL or FAC2MF2K. The first of these programs applies the kriging factors to generate a MODFLOW-compatible real array. Pilot point values can be read from the same file as read by PPK2FAC, or they can be read from a different pilot points file as long as that file cites the same pilot points in the same order, and as long as pilot points are assigned to the same zones as in the original pilot points file read by PPK2FAC. If FAC2REAL is used to build a MODFLOW input array as part of a composite model run by PEST for the purpose of estimating hydraulic properties at the sites of pilot points, a PEST template file can be built from the pilot points file. Prior to each model run PEST will build a new pilot points file from this template file for the use of FAC2REAL. FAC2REAL will then build a real array in which the latest parameter values, as calculated by PEST, are interpolated to model grid cell centres prior to being read by MODFLOW.

FAC2MF2K is used to modify a set of MODFLOW-2000 input files in order that pilot point parameterisation can be undertaken using MODFLOW-2000. In this case PPK2FAC-generated kriging factors are used to build MODFLOW-2000 *multiplier* arrays. Note, however, that where interpolation is based on log variograms, rather than native variograms, the USGS version of MODFLOW-2000 cannot be used to calculate hydraulic property arrays using pertinent parameter values and *multiplier* arrays for, as presently coded, it cannot undertake array-generation based on the logs

of parameters. Instead, a modified version of MODFLOW-2000 named MF2KASP must be used. See Section 5 of Part A of this manual for further details.

If undertaking pilot-point-based parameter estimation using PEST, it is often desirable that PEST work in regularisation mode rather than parameter estimation mode in order that stability of the parameter estimation process can be guaranteed, and in order that geologically reasonable parameter values are estimated. A PEST control file which uses pilot-point-based parameters can be modified using program PPKREG to include regularisation prior information based on the contents of a regularisation information file generated by PPK2FAC.

**See Also**

See also FAC2MF2K, FAC2REAL, FIELDGEN and PPKREG.

# PPK2FACF

**Function of PPK2FACF**

PPK2FACF is one member of a group of programs which expedites the use of pilot points as a device for spatial parameterisation of a MicroFEM model. Other members of this suite are FAC2FEM, PPKREG and FEM2SMP. PPK2FACF and FAC2FEM are very similar to PPK2FAC and FAC2REAL respectively which implement the use of pilot points in the MODFLOW/MT3D context. PPKREG is used in both the MODFLOW/MT3D and MicroFEM contexts. FEM2SMP converts MicroFEM output data to bore sample file format so that time-interpolation and PEST control file construction utilities used in the PEST-MODFLOW interface (viz. SMP2SMP and PESTPREP) can also be used with MicroFEM.

PPK2FACF generates a set of kriging factors for use in spatial interpolation from a set of pilot points to a MicroFEM mesh. Kriging factors are based on user-supplied, nested variograms, each with an arbitrary magnitude and direction of anisotropy. Different variograms can be used for spatial interpolation in different parts of the model domain. PPK2FACF also writes a "regularisation information file" which can be used to introduce geostatistically-based regularisation constraints to a parameter estimation problem.

Generation of MicroFEM input files based on PPK2FACF-generated kriging factors is carried out by FAC2FEM. Separation of the time-consuming, factor-generation process from the spatial interpolation process facilitates automatic parameter estimation based on pilot points using software such as PEST, for kriging factors are unchanged as values assigned to the pilot points, and then to the mesh through spatial interpolation, are adjusted through the parameter estimation process.

Regularisation information recorded by PPK2FACF is used by program PPKREG; PPKREG modifies an existing PEST control file containing pilot-point-based parameters, adding regularisation constraints based on the same geostatistical structures as those used for the generation of kriging factors.

**Running PPK2FACF**

PPK2FACF is very similar to PPK2FAC. Only differences between these two programs will be described; the user should refer to the documentation of PPK2FAC for an explanation of the salient aspects of the use of PPK2FACF.

When using PPK2FACF, don't forget that if you respond to any of its prompts by simply typing "E" or "e" and then <Enter>, you can backtrack to the previous prompt. The same applies to all members of the Groundwater Data Utilities.

Upon commencement of execution PPK2FACF asks the user for the name of a pilot points file, which it then proceeds to read. However, unlike PPK2FAC, PPK2FACF does not need to read a grid specification file, this file being useful only in the MODFLOW/MT3D context for conveying the size, disposition and design of the

finite difference grid upon which these latter models are based. Instead of the design of a finite difference grid, PPK2FACF must acquaint itself with the design of a finite element mesh. It also needs to know the zonation of this mesh. (Recall that pilot points can be assigned to different zones in a pilot points file; interpolation for each zone within a finite element mesh takes place only on the basis of pilot points assigned to that zone.)

To inform itself of mesh design and zonation PPK2FACF prompts for the names of two MicroFEM files, these being a "fem" file and a "special ASCII label" file. The respective prompts are:-

```
Enter name of fem file defining finite element mesh:
Enter name of special ASCII label file defining zonation:
```

PPK2FACF reads the number of nodes in the mesh from the first of these files. It reads node coordinates and zone numbers from the second of these files. It is the user's responsibility to ensure that this file meets the expectations of PPK2FACF; an example follows:-

```
0.000 6000.000 2
214.286 6000.000 2
0.000 5777.778 2
428.571 6000.000 2
315.317 5822.733 2
0.000 5555.556 2
242.093 5616.571 2
642.857 6000.000 3
518.925 5854.790 3
486.844 5674.949 3
0.000 5333.333 3
214.240 5403.349 3
434.535 5463.636 3
857.143 6000.000 4
718.385 5780.881 4
684.675 5497.692 4
0.000 5111.111 4
204.523 5188.182 4
402.321 5261.606 4
571.162 5322.899 4
929.639 5826.084 4
```

**Part of a special ASCII label file.**

A special ASCII label file has three columns of data; the first two columns are comprised of the eastings and northings of all nodes within the finite element mesh, while the third column is comprised of zone numbers. This file should be prepared within MicroFEM as follows:-

1. Create a label field for the mesh (select "Project Manager", then "Unit", then "Add", then "Labels" with the "New" radio button selected).

2. In "walking mode" mark different sections of the mesh according to the desired zonation; **provide only integer values to the label field**. Assign these integer values to marked or unmarked areas of the mesh while in MicroFEM "input mode". Make sure that **every node** is assigned an integer in this manner.

3.  Select "Export" from the main menu, then "Special ASCII files" while the respective label field is highlighted in the bottom right of the MicroFEM window. The file type is "&.."; save the file, giving it a suitable name.

The saved special ASCII file should have a similar format to that depicted in the above figure. **It is important to ensure that every line of this file has 3 entries; this will only occur if a label is assigned to every node in the mesh.** For those zones in which pilot point interpolation is to take place, the zone number must correspond to a zone number appearing in the pilot points file upon which spatial interpolation will be based. In the simplest case where only one zone is used, all nodes will be assigned the same integer value; this will be the same zone number to which all points in the pilot points file are assigned.

Other aspects of PPK2FACF use are very similar to those of PPK2FAC. Like PPK2FAC, PPK2FACF requests the name of a structure file, and the names of the structural elements to be used for spatial interpolation within each zone. See the documentation of PPK2FAC for details.

Like PPK2FAC, PPK2FACF writes a (formatted or unformatted) factor file and a regularisation information file; the latter file is used by PPKREG for adding regularisation information to a PEST control file. Unlike PPK2FAC however, PPK2FACF does not write a standard deviation array file.

**Uses of PPK2FACF**

The factor file written by PPK2FACF is used by FAC2FEM to write a MicroFEM input file in which a user-nominated aquifer property is calculated by spatial interpolation from a set of pilot points. Whether or not MicroFEM is being used in conjunction with PEST for pilot-point-based calibration, this can provide an extremely useful means of spatial interpolation from a set of measurement or data points.

Where pilot-point values are assigned through calibration, regularisation should be introduced to the calibration process because of the large number of parameters that normally require estimation. As is explained elsewhere in this manual, the more pilot points that are used for spatial parameterisation, the less likely is the resulting parameter field to be "blotchy", and the less likely it is that one or a number of pilot points will be assigned unusually high or low values. Regularisation does much to ensure that realistic parameter fields will result from the calibration process, and reduces the likelihood of numerical instability. Regularisation information can be added to a PEST control file using PPKREG; in doing this, PPKREG uses the "regularisation information file" written by PPK2FACF. See the documentation of PPK2FAC for further details.

**See Also**

See also FEM2SMP, FAC2FEM and PPKREG.

# PPK2FAC1

**Function of PPK2FACF1**

PPK2FAC1 is identical to PPK2FAC except for the fact that it writes a regularisation data file which contains more information than the corresponding file written by PPK2FAC. Such a file is suitable for the use of PPKREG1, and enhanced version of PPKREG.

# PPKREG

**Function of PPKREG**

Where a PEST control file has been constructed to estimate hydraulic property values at pilot point locations, regularisation information can be automatically incorporated into the parameterisation process using PPKREG. This information is added to the PEST control file in the form of a set of prior information equations pertaining to preferred differences between property values (or their logs) assigned to the pilot points. The weight assigned to each of these prior information equations can be the same for each, or it can be calculated according to the separation between the respective pilot points. In the latter case the weight is calculated as proportional to the inverse of the nested variogram pertaining to the zone in which the pilot points are situated. Geostatistical information by which this calculation is made is read from a "regularisation information file" produced by program PPK2FAC.

The use of pilot points for groundwater model parameterisation is often accompanied by a tendency to over-parameterise the problem (ie. to attempt to estimate property values at more pilot point locations than it is within the capacity of the calibration dataset to achieve with any degree of uniqueness). This is not necessarily a bad thing, because one of the benefits of using pilot points for spatial parameterisation is that a superfluity of these points can allow PEST to determine the locations at which geological heterogeneity must be introduced in order to achieve a good fit between model outcomes and corresponding field measurements. However in order for this process to work, it is essential that regularisation constraints (normally in the form of "homogeneity constraints" or "smoothness constraints") be incorporated into the parameter estimation process. This is most easily achieved using PPKREG.

**Using PPKREG**

*Prior to Using PPKREG*

A number of conditions must be satisfied if PPKREG is to be used for introducing regularisation information into a parameter estimation problem. These are as follows:-

1. The PEST control file to which regularisation constraints are to be introduced must include one or a number of different types of parameters linked to a set of pilot points. These parameter types can represent different hydraulic properties, or the same hydraulic property in different model layers.

2. The name of each parameter that is linked to a pilot point must include the name of the pilot point (as named in the pilot points file) to which it is linked. Each parameter of a particular type must have the same prefix (or no prefix at all), followed by the pertinent pilot point name. The prefix can be up to two characters in length.

3. A PPKREG run must be preceded by a PPK2FAC run. Normally this will have been done as a matter of course, for the interpolation factor file generated by PPK2FAC will be required by FAC2REAL for array generation as part of a composite model run by PEST, or will be used by FAC2MF2K to modify a MODFLOW-2000 dataset such that it includes pilot-point-based parameters. (The MODFLOW-2000 input dataset can then be translated to PEST format using the MODFLOW2000-to-PEST translator, MF2PEST.) The regularisation information file generated by PPK2FAC is then used by PPKREG for adding prior information to the pilot-point-based PEST control file.

4. Where different parameter types in a PEST control file pertain to different sets of pilot points, PPKREG must be run twice in order to add different sets of regularisation prior information to the PEST control file based on the different sets of pilot points.

Note that it is not essential that every parameter cited in a PEST control file be linked to a pilot point. Those parameters which are not linked to a pilot point are simply omitted from any regularisation information that is introduced to the PEST control file by PPKREG. However care must be taken where parameters are omitted from inclusion in regularisation constraints; as is explained in the documentation to PEST-ASP, if regularisation is introduced to a parameter estimation problem in order to render it numerically more tractable, it may not live up to expectations unless all parameters (or at least the vast majority or parameters) are included in the regularisation process.

*Running PPKREG*

PPKREG begins execution by prompting for the name of an existing PEST control file:-

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

Supply a name as appropriate. PPKREG's task is to write a new PEST control file based on this existing PEST control file; in the new control file PEST is asked to run in regularisation mode. Regularisation constraints in the form of prior information will be included in this new PEST control file, and a "regularisation" section will be appended to the end of the file. If any prior information is present within the original PEST control file, this will be included in the new PEST control file. However an existing "regularisation" or "predictive analysis" section will be ignored.

PPKREG next prompts for the name of a PPK2FAC-generated regularisation information file. Its prompt is:-

```
 Enter name of regularisation information file:
```

As is explained in the documentation to PPK2FAC, this file lists the names of the pilot points contained in the pilot points file upon which PPK2FAC based its calculation of kriging factors; it also contains nested variogram values for each pair of pilot points that are jointly used for interpolation to at least one cell centre. This latter

condition limits the provision of regularisation information to points which are in the same zone, and which are "close enough together", as defined by the search radius or "maximum number of points for interpolation" criterion supplied to PPK2FAC when it generated the kriging factors. Hence these variables can be used to limit the number of prior information equations added by PPKREG to an existing PEST control file. Basically, the smaller is the search radius, and the fewer the maximum number of points used for interpolation, the fewer will be the number of prior information equations containing regularisation information added to the PEST control file. Often PPK2FAC will be run only once in the course of preparation for pilot-point-based parameterisation. In this case the same variables used in the calculation of kriging factors will be used in the calculation of regularisation information. However the modeller is free to undertake a separate PPK2FAC run (using a different search radius and/or specifying a different value for the maximum number of points used for interpolation) for each of these separate purposes if he/she feels that this is warranted.

Next PPKREG prompts:-

```
How many pilot-point-based parameter families in PEST control file
pertain to this regularisation information file:
```

As was discussed above, each type (or family) or pilot-point-based parameters featured in the one regularisation information file must have its own parameter name prefix of up to two characters in length (or no prefix at all if desired). The remainder of the name of each parameter must be the same as the name of the pilot point that it represents. Note also, that a PEST control file may contain different sets of pilot points pertaining to different hydraulic properties; the regularisation information pertaining to each of these sets may reside in different regularisation information files. PPKREG will then need to be run for each set separately. This is a perfectly acceptable procedure; the user only needs to ensure that pilot points are named according to different protocols within the different pilot point groups whose regularisation information resides, in turn, in different regularisation information files.

For each family of pilot-point-based parameters pertaining to the one regularisation information file, PPKREG asks the following questions:-

```
For family number n:-
   Enter parameter prefix (<Enter> if none):
   Apply uniform or geostatistical regularisation? [u/g]:
   Enter weight multiplier:
   Enter new regularisation group name:
   Enter root name for new prior information:
```

The answer to the first of the above questions serves to distinguish one pilot-points-based parameter set from another where both are based on the same set of pilot points for which regularisation is contained within the one pilot points file; supply the prefix as appropriate.

For each pair of points for which a variogram value is supplied in the regularisation information file, PPKREG generates one prior information equation. This equation expresses the fact that the difference between the corresponding parameter values (ie. the hydraulic property values assigned to these pilot points) is zero; if the parameters are log-transformed in the "parameter data" section of the PEST control file, then the

pertinent prior information equation states that the difference between the logs of the respective parameter values is zero. The weight applied to each such item of prior information can be the same ("u" option in answering the second of the above prompts). Alternatively, the weight can be calculated as the inverse of the square root of twice the magnitude of the nested variogram pertaining to the two points comprising the pair. This method of calculating the weight is in harmony with the fact that the square root of twice the variogram is equivalent to the standard deviation of the parameter difference, if the statistical dependence of this difference on point separation, as characterised by the variogram, is correct.

The third of the above prompts requires that a weight multiplier be supplied. If uniform weights assignment is undertaken, this is the value of the weight assigned to all articles of regularisation prior information. If geostatistically-based weights are calculated, this acts as a multiplier (applicable to all regularisation prior information equations) for weights calculated on the basis of variogram values in the manner described above.

PPKREG assigns regularisation information to a "regularisation subgroup". Since version 6.0, PEST has allowed the existence of multiple regularisation groups within the one PEST control file. The name of each such group must begin with the characters "regul". In the fourth of the above prompts, PPKREG asks the user for the name of the group to which the new pilot-point-based regularisation equations should be assigned. This name must begin with "regul" and must be different to the name assigned to any other regularisation subgroup within PPKREG, or already existing within the PEST control file. (Note that PEST can adjust the relative weighting attached to different regularisation subgroups during the inversion process; this is done if the regularisation control variable IREGADJ is set to 1. When writing the "regularisation" section of the PEST control file, PPKREG supplies a value of 1 to IREGADJ, thus asking PEST to implement this automatic inter-group weights assignment procedure. If this is not desired, it can be rectified by direct editing of the PEST control file.)

Finally PPKREG prompts for the root name of the prior information equations that it writes to encapsulate the regularisation information. Supply a string of 6 characters or less; PEST will affix an equation number to this string when naming prior information equations. The user should ensure that the resulting prior information name does not conflict with any names that may already be represented within the existing PEST control file to be modified by PPKREG (PESTCHEK will soon inform you if this condition has been violated).

Next PPKREG searches for linkages between parameters and pilot points; as was discussed above, parameters and pilot points are linked through their names. Parameters not linked to any pilot points are listed to the screen; PPKREG asks the user to verify that it is alright to continue execution, even though not all of the parameters cited in the PEST control file are pilot-point-based. (When running PPKREG multiple times in order to sequentially add regularisation information pertaining to more than one regularisation group, the list of unlinked pilot points on any one PPKREG run may be very large; do not be perturbed by this.)

Next PPKREG prompts for the name of the new PEST control file which it must write:-

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

and, finally, for some information to use in the "regularisation" section of this file:-

```
 Enter target measurement objective function PHIMLIM:
 Enter initial regularisation weight factor WFINIT:
 Enter min. reg. weight factor WFMIN ( <Enter> if 1.0000E-10):
 Enter max. reg. weight factor WFMAX ( <Enter> if 1.0000E+10):
```

As is explained in the documentation to PEST-ASP, the first of these quantities should be set slightly above the objective function minimum that it is possible to achieve without the imposition of regularisation constraints. Alternatively, if this is not known, it should be set at some reasonable value calculated from an anticipated, or acceptable, level of model-to-measurement misfit. The initial weight factor WFINIT (see the second of the above prompts) can mostly be set to 1. Default upper and lower weight factor bounds, set at $10^{10}$ and $10^{-10}$ times the initial weight factor, are supplied in the pertinent prompts for these quantities by PPKREG. These can be accepted by simply pressing the <Enter> key in each case. However the user should monitor the weight factors calculated by PEST as the optimisation process progresses in order to establish whether these bounds should be set wider or narrower.

*Warnings and Error Conditions*

Use of PPKREG is quite straightforward. However a few simple rules should be observed. Where PPKREG notices any violations of these rules it will issue an appropriate error or warning message; however in some cases it may not be aware that a rule has been violated, in which case a warning cannot be issued. The rules are listed hereunder.

1. If kriging for a particular parameter type is based on a geostatistical structure (as recorded in a structure file read by PPK2FAC) in which the TRANSFORM type is "log", then all parameters of this type should be log-transformed in the parameter estimation process. Thus any regularisation information introduced to the PEST control file based on that parameter type will pertain to the logs of the pertinent parameter value differences rather than to the parameter value differences themselves. If the geostatistical option is selected for weights calculation, then these weights will be based on a nested variogram that depicts the logarithmic nature of the geostatistical structure pertaining to that parameter. Similarly, if TRANSFORM is set to "none" for the pertinent geostatistical structure, and regularisation information is geostatistically weighted, then the parameter should not be log-transformed in the parameter estimation process. Note, however, that neither PPKREG nor PEST will object if these rules are violated, for they have no knowledge of the geostatistical structure underlying weights calculation.

2. PPKREG will object, however, if an attempt is made to impose a regularisation constraint on the difference between the values assigned to

two pilot-point-based parameters if one is log-transformed in the existing PEST control file and the other is not.

3. If there are any parameters in the existing PEST control file which are part of a pilot point family, but which are not linked to any other pilot points through regularisation information contained in the regularisation information file, PPKREG will list such points to the screen and ask if this is alright. This can happen if, for example, there is only one pilot point assigned to a particular zone (in which case the zone is effectively homogeneous), or if the search radius used when running PPK2FAC was not large enough. If you ignore this warning, you can add pertinent prior information regularisation constraints pertaining to this parameter to the PEST control file yourself if you wish.

4. If there are any pilot-point-based parameters that are linked to only one other pilot-point-based parameter through regularisation prior information, then PPKREG will issue a similar warning.

5. If a parameter is tied to another parameter, it cannot be included in prior information. Hence all regularisation information linking such a parameter to any other parameters in the regularisation information file is ignored.

6. The same holds for fixed parameters.

**Uses of PPKREG**

As has already been described, PPKREG facilitates the introduction of prior information to a PEST control file in which parameters are based on pilot points. Use of this prior information in the parameterisation process attempts to ensure that estimated parameters adhere to the geostatistical structure of an area where this is known. Different families of pilot points parameters can be used for different parameter types. These can be based on the one set of pilot points (in which case a single PPKREG run can be used to add regularisation information pertaining to all such parameters) or on different sets of pilot points for which regularisation is stored in multiple regularisation information files (in which case multiple PPKREG runs will be required).

PPKREG is most useful where MODFLOW/MT3D-based parameterisation is carried out using FAC2REAL in conjunction with MODFLOW/MT3D, or where a MODFLOW-2000 input dataset was built with the help of FAC2MF2K, the latter program being used to introduce pilot-point-based parameters to an existing MODFLOW-2000 input dataset. In the latter case, the set of MODFLOW-2000 input files must be translated to PEST format using the MODFLOW2000-to-PEST translator, MF2PEST. Once this has been done, regularisation information can be added to the pertinent PEST control file using PPKREG using the instructions provided herein.

**See Also**

See also FAC2MF2K, FAC2REAL and PPK2FAC

# PPKREG1

**Function of PPKREG1**

PPKREG1 is an advanced version of PPKREG. Like PPKREG, it is used to add prior information to a PEST control file, with this prior information encapsulating regularisation constraints on the inversion process; the purpose of these constraints is to provide a "preferred system condition", the use of which adds numerical stability to the inversion process. However PPKREG1 provides more regularisation alternatives than PPKREG, the latter only providing a "smoothing regularisation" option in which a set of prior information equations is provided expressing the fact that the difference between pairs of parameter values (or the logs of these values) is preferentially zero. PPKREG1 provides the user with the option of using an alternative form of regularisation referred to as "preferred value" regularisation herein. In implementing this form of regularisation, prior information equations are provided in which each parameter (or its log) is assigned a preferred value. A single such value can be assigned to all parameters, or values can be assigned individually. A single weight can then be assigned to each pertinent prior information equation, or a covariance matrix can be assigned to the family of prior information equations based on a geostatistical characterisation of parameter variation.

PPKREG1 allows the user to adjust regularisation weights (or components of the regularisation covariance matrix) in accordance with data density. Thus greater weights can be assigned to prior information equations pertaining to parameters that are far removed from observation points in comparison to prior information equations which cite parameters that are relatively close to observations points.

**Running PPKREG1**

*General*

PPKREG1 has much in common with PPKREG. The discussion below focuses on the differences between these two programs; see documentation of PPKREG for a discussion of functionality that is common to both of them.

PPKREG1 execution is initiated by typing its name at the command-line prompt. Upon commencement of execution it prompts for the name of an existing PEST control file. PPKREG1's task is to add prior information, and a regularisation section, to this PEST control file. If the existing PEST control file already possesses a regularisation section, this section will be overwritten.

After reading the existing PEST control file, PPKREG1 prompts:-

```
Enter name of regularisation information file:
```

This file will have been written by PPK2FAC1 (not PPK2FAC, which is superseded by PPK2FAC1). FAC2REAL calculates kriging factors for spatial interpolation from a set of pilot points to the model grid for the purpose of constructing a MODFLOW-compatible real array. It also records geostatistical information pertaining to the

physical property values which are sampled at pilot point locations. This information is based on the contents of the structure file read by PPK2FAC1, and on the user's responses to prompts issued by PPK2FAC1 during its execution.

Next PPKREG1 asks:-

```
How many pilot-point-based parameter families in PEST control file
pertain to this regularisation information file:
```

Supply a number no lower than unity; as is explained in the documentation to PPKREG, each set of parameters which is based on the same set of pilot points must have a different parameter prefix (which can be the "null prefix" if desired) affixed to the names of respective pilot points.

PPKREG1 now issues a series of prompts for each family of pilot point parameters that pertain to the nominated regularisation information file. The first of these prompts requests the name of the prefix which characterises that parameter family:-

```
Enter parameter prefix (<Enter> if none):
```

*Smoothness Regularisation*

PPKREG next prompts for the type of regularisation to which that family of parameters will be subject:-

```
Apply smoothness or preferred value regularisation? [s/p]:
```

If you respond with an "s" to the above prompt, PPKREG1 will write the same set of regularisation prior information equations as those written by PPKREG; see documentation of PPKREG for details. Furthermore, like PPKREG, PPKREG1 will allow the user to chose between uniform and geostatistical weighting for these prior information equations:-

```
Use weights of unity or geostatistical weighting? [u/g]:
```

Note that if the unity option is selected, all weights are temporarily assigned a value of 1.0 (this can be altered shortly through the use of a weight multiplier). Alternatively, selection of the geostatistical option allows weights to be calculated from the hydraulic property variograms assigned to this area as described in the documentation to PPKREG. In this case prior information equations (expressing preferred parameter equality) pertaining to points which are close together are weighted more heavily than those pertaining to points which are further apart.

A weight multiplier can be applied to prior information weights calculated according to the previous prompt. A single weight multiplier can be used for all new prior information equations pertaining to a particular parameter family. Alternatively, prior information weights can be calculated in accordance with proximity of pilot points to data points (ie. points at which data is available for use in the calibration process). PPKREG1 prompts:-

```
Use uniform or data-density-dependent weight multiplier? [u/d]
```

If the user requests a uniform weight multiplier, that multiplier is requested next:-

```
Enter uniform weight multiplier:
```

Alternatively, if data-density-dependent weights are requested, PPKREG1 prompts for the name of a "data coordinates file". This file must have at least three data columns; PPKREG1 reads data point eastings from the second column and data point northings from the third column. Thus a pilot points file and a bore coordinates file both satisfy the requirements of a data coordinates file.

Note that where more than one pilot-point-based parameter family is managed by PPKREG1, PPKREG1 still prompts for only one data file. It is assumed that if data-density-dependent weighting is used for both of these families, data density is the same for each family.

Each prior information equation used to implement smoothing regularisation cites two parameters. For each one of these parameters a notional weight multiplier *w* is calculated using the equation:-

$$w = a + b \sum_{i=1}^{n} r_i^c$$

where *a*, *b*, *c* and *n* are supplied by the user, and *r* is the distance between the pilot point associated with that parameter and data points cited in the data coordinates file. (Note that summation in the above equation takes place over the *n* nearest data points to the current pilot point; it is suggested that *n* be supplied as 1.) The weight multiplier calculated for the entire prior information equation is then calculated as the geometric average of the two individual parameter weight multipliers. PPKREG1's prompts are:-

```
Enter a:
Enter b
Enter n:
Enter c:
Enter maximum allowable weight factor:
Enter minimum allowable weight factor:
```

Note the last two of the above prompts in which the user may assign a minimum and maximum notional weight factor to each pilot-point-based parameter; this assignment takes place before geometric averaging of notional parameter weight factors to calculate the weight factor assigned to the entire prior information equation.

*Preferred Value Regularisation*

If preferred value regularisation is selected for a particular parameter family, PPKREG1 prompts:-

```
Use uniform preferred value, or read it from a file? [u/f]:
```

If the "uniform" option is selected, the user must provide the preferred value for all pilot point parameters within the parameter family in response to the prompt:-

```
Enter uniform preferred value:
```

Alternatively, if preferred values are to be read from a file, PPKREG1 prompts for the name of a pilot points file; pertinent parameter values are read from the fifth column of this file. Note that the first column of this file must contain pilot point names rather than parameter names; the latter will differ from the former where a parameter prefix is deployed.

As in the case of smoothness regularisation, PPKREG1 next prompts:-

```
Use weights of unity or geostatistical weighting? [u/g]:
```

In the latter case, individual weights are not calculated at all; rather a covariance matrix is supplied for prior information pertaining to this parameter family. Each prior information equation used to implement preferred value regularisation cites a single parameter, assigning a preferred value to that parameter. If a covariance matrix is assigned to that family of prior information equations, the dimensions of that matrix are the same as the number of prior information equations introduced to the PEST control file for that parameter family by PPKREG1. The elements of the covariance matrix are calculated from geostatistical information contained in the regularisation information file, and are thus based on the variogram(s) supplied to PPK2FAC1 when it wrote this file. Note that the range of inter-parameter correlation is limited by the interpolation search distances, and number-of-pilot-point limits supplied by the user to PPK2FAC1 when providing kriging specifications; if it is desired that kriging and regularisation be governed by different variograms, or that different variables govern the use of these variograms when deployed for these separate processes, PPK2FAC1 should be run twice – once to calculate kriging factors and once to generate the regularisation information file to be used by PPKREG1.

As in the case of smoothness regularisation, a uniform or data-density-dependent weight multiplier can be assigned to prior information introduced to the PEST control file by PPKREG1:-

```
Use uniform or data-density-dependent weight multiplier? [u/d]:
```

As is described above, data-density-dependent weight multipliers are calculated on the basis of data point coordinates supplied in a data coordinates file. Where geostatistical weighting is employed, and thus a covariance matrix is used instead of weights, each entry of the covariance matrix is divided by the product of the weight multipliers associated with the individual pilot points to which the entries pertain. In this manner regularisation constraints are more strongly enforced where parameter pilot points are closer to data points than where they are not. Where a uniform weight multiplier is requested, all elements of the covariance matrix are divided by the square of this multiplier.

Where preferred value regularisation is employed and the geostatistical weights option is selected, PPKREG1 also prompts for the name of a file in which to store the covariance matrix for the newly introduced family of prior information equations:-

```
Enter covariance matrix file for prior information:
```

This file is written by PPKREG1, while the name of this file is cited in the "observation groups" section of the new PEST control file written by PPKREG1. Note that where a covariance matrix is employed, weights of 1.0 are assigned to all prior information equations; as is documented in the PEST manual, these weights are ignored by PEST.

*Continuing PPKREG1 Execution*

For each new family of prior information equations introduced to the PEST control file, PPKREG1 prompts for the name of an observation group to which this prior

information is to be assigned. This is referred to as a "regularisation group" by PPKREG1, for it is assumed that all new prior information introduced to the PEST control file by PPKREG1 will be used for regularisation purposes. The name must begin with the string "regul" in accordance with PEST's regularisation conventions. The name must also be different from that assigned to any other observation group, either within the existing PEST control file, or supplied to PPKREG1 for other prior information families during its current run. PPKREG1 also prompts:-

```
Enter root name for new prior information:-
```

The name of each new prior information equation introduced to the PEST control file by PPKREG1 is formed through appending this root name to the front of a number; this number is formed by counting (from 1) new prior information equations introduced for each family.

PPKREG1 next prompts for the name of the PEST control file which it must write; this name must be different from that of the PEST control file from which it reads data initially. Then PPKREG1 prompts for the values of a number or regularisation variables; default values are supplied for other regularisation variables:-

```
Enter target measurement objective function PHIMLIM:
Enter initial regularisation weight factor WFINIT:
Enter min. reg. weight factor WFMIN ( <Enter> if 1.0000E-10):
Enter max. reg. weight factor WFMAX ( <Enter> if 1.0000E+10):
```

PPKREG1 then writes the new PEST control file and any covariance matrix files which need to be written; then it ceases execution.

*PPKREG1 Warnings*

In the course of its execution PPKREG1 may issue a number of warnings, and prompt the user whether it is alright to proceed. Conditions giving rise to these warnings include the following:-

1. A certain pilot point has no geostatistical linkage to any other pilot points cited within the regularisation information file read by PPKREG1. This can occur if a single pilot point exists within a certain model zone, and/or if the interpolation search radius supplied by the user to PPKFAC1 is too short.

2. One or more pilot point parameters are fixed or tied. In this case PPKREG1 will add no prior information to the PEST control file pertaining to these pilot points.

3. Some parameters exist within the PEST control file which are not linked to the set of pilot points for which regularisation information is supplied in the regularisation information file.

**Uses of PPKREG1**

Uses of PPKREG1 are the same as those of PPKREG; however it allows the implementation of more complex regularisation options than does PPKREG. PPKREG1 can be used as a replacement for PPKREG, for it contains all of the options that are available through this program. Through sequential use of VERTREG

and PPKREG1 the user is able to introduce a set of complex regularisation conditions to a PEST control file whose parameterisation is based on one or a number of pilot point families occupying one or a number of model layers. Different sets of prior information equations introduced through this process should be assigned to different observation groups. When undertaking regularised inversion, the relative weighting assigned to each of these groups may sometimes be difficult to determine, and a trial and error process may be required to determine the best set of weights. The WTFACTOR utility (part of the PEST suite) may be useful in implementing this process. Alternatively, PEST may be able to adjust relative regularisation weights automatically; the IREGADJ variable may be of use here. PEST's "adaptive regularisation" capabilities may also be of use.

**See Also**

See also PPKREG, PPK2FAC, PPK2FAC1 and VERTREG.

# PT2ARRAY

**Function of PT2ARRAY**

PTARRAY constructs a MODFLOW/MT3D-compatible real array from the data contained in a bore information file. It first determines which bores cited in the bore information file lie in which cells. Then it assimilates all information pertaining to each cell before assigning an appropriate value to the cell's corresponding array element either by direct addition or by addition and subsequent cell area normalisation.

**Running PT2ARRAY**

A settings file `settings.fig` must be present in the directory from which PT2ARRAY is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

PT2ARRAY commences execution with the prompt:

```
 Enter name of grid specification file:
```

to which you should respond by entering the appropriate filename. If a default filename for the grid specification file has been read from a filename file (`files.fig`) resident in the current directory, that filename will appear with the above prompt. It can be accepted through pressing the <Enter> key or rejected by supplying the correct filename.

Next PT2ARRAY prompts:

```
 Enter name of bore coordinates file:
```

Once again, a default bore coordinates filename may be supplied with above prompt, depending on the existence and contents of a filename file in the current directory. Note that it is not necessary that the bore coordinates file read by PT2ARRAY possess a final column containing layer numbers. If this final column is present, PT2ARRAY ignores it.

PT2ARRAY next requests the name of a bore information file; each bore cited in the bore information file must also be referenced in the bore coordinates file whose name was provided in response to the previous prompt. Section 2.4 of Part A of this manual discusses bore information files in detail. It is the function of PT2ARRAY to extract the information contained in one of the columns of the bore information file and insert it into the cells of a MODFLOW/MT3D-compatible real array, the insertion cell for a particular item of information being the cell containing the bore to which that item of information pertains. PT2ARRAY prompts:

```
 Enter name of bore information file:
```

to which you should respond by supplying an appropriate filename. Next you must identify the column within this bore information file containing the data which PT2ARRAY must incorporate into the real array.

```
 Use which column of bore information to generate real array:
```

Enter the number of any column but the first. Do not enter a number greater than the number of columns represented in the bore information file.

For each cell of the finite difference grid, PT2ARRAY determines which bores from the bore information file lie within that cell. It then adds together all the data (as found in the relevant column of the bore information file) pertaining to those bores which occupy the same cell. If desired, this data sum can be converted to another system of units, or from volumes to rates etc, through the use of a multiplication factor:

```
 Enter multiplication factor for data in this column  [1.00]:
```

Enter a suitable number or press <Enter> if no conversion factor is required. If desired, the cumulative information for a certain cell can also be divided by the area of that cell before being assigned to the real array:

```
 Divide by cell area?  [y/n]
```

PT2ARRAY assigns values to all real array elements for which the corresponding model cells contain one or more bores. Real array elements for which cells have no bores are assigned a value of zero. PTARRAY then prompts for the filename to which it should write the new real array:

```
 Enter name for real array output file:
```

Supply a filename in response to this prompt, remembering to observe the filename extension convention for formatted and unformatted real array files discussed in Section 2.17 of Part A of this manual.


**Uses of PT2ARRAY**

In many MODFLOW modelling applications it is useful to combine recharge and pumping into a single array, supplied to the model as a recharge array. Note that this process will not be possible for a multiple aquifer model where water is extracted from a different model layer to that which receives recharge. However where it is possible, the combination of all extracted and added water for each stress period into a single array may render model preprocessing easier, especially if pumping and recharge determination are carried out by external software and provided to the model just before runtime.

The construction of the pumping component of such an array (which may be combined with the recharge component using program TWOARRAY) is easily achieved with the help of PT2ARRAY. Using program PMP2INFO, a bore information file containing historical pumping figures can be constructed for a

particular stress period. PT2ARRAY can then be used to incorporate these pumping figures into the model. A suitable multiplication factor must be supplied to convert the pumped figures from the bore information file into pumping rate figures for the use of the model; note that the factor must be negative to account for the fact that pumping data is being supplied as part of a recharge array. Also, as recharge is normally supplied as a rate per unit area, pumping rates supplied as part of a recharge array must be divided by the cell area, an operation that is supported by PT2ARRAY.

When using PMWIN as a MODFLOW preprocessor, PT2ARRAY can still be used for pumping (and other) data assimilation even where the latter is provided to MODFLOW in its usual tabular form. This is made possible by the fact that PMWIN is able to import listed data types such as pumping data, drainage data, etc in array format, with zero values assigned to unaffected cells.

**See Also**

See also PMP2INFO, PTINGRID.

# PTINGRID

**Function of PTINGRID**

PTINGRID locates points (for example bores) with respect to the finite difference grid. It obtains point coordinates from a bore coordinates file and grid specifications from a grid specification file. It calculates the cell in which each point (as read from a bore listing file) lies and, optionally, records the value of a MODFLOW/MT3D-compatible integer or real array pertaining to that cell.

**Using PTINGRID**

A settings file `settings.fig` must be present in the directory from which PTINGRID is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

Upon commencement of execution PTINGRID prompts:

```
 Enter name of grid specification file:
```

Enter an appropriate filename. Alternatively, if a filename file (`files.fig`) containing a default grid specification filename is present in the current directory, PTINGRID displays the default grid specification filename with the above prompt; press <Enter> to accept the default or type in the more appropriate filename.

Next PTINGRID requests the name of a bore coordinates file:

```
 Enter name of bore coordinates file:
```

If read from a filename file, a default bore coordinates filename will be included in the above prompt; accept the default with <Enter> or reject it by typing in the correct filename. In most cases information on grid location will be required for only a subset of the points appearing in the bore coordinates file. Hence PTINGRID requests the name of a bore listing file in order to ascertain the names of those points for which grid location information is desired:

```
 Enter name of bore listing file:
```

Provide the name of an appropriate bore listing file. Alternatively, provide the name of the bore coordinates file again if information on point location with respect to the grid is required for all bores cited in that file.

PTINGRID is able to provide, in addition to the cell in which each point lies, the value of a MODFLOW/MT3D-compatible real or integer array for the cell containing each point. So at this stage PTINGRID prompts the user for the name of a file holding such an array. Respond with "n" if you do not wish to activate this PTINGRID facility.

```
 Read integer array, real array or none [i/r/n]:
```

If you respond with "i" or "r" PTINGRID next prompts for the name of the file holding the integer or real array and, if necessary, whether the file is formatted or unformatted. See Sections 2.10 and 2.17 of Part A of this manual for the naming conventions associated with integer and real array formatted and unformatted files. Use of one of the standard extensions will result in PTINGRID assuming a formatted or unformatted file as appropriate; if a standard extension is not used, PTINGRID must prompt the user specifically for the file's status in this regard.

```
 Enter name of real array file:
 Is this a formatted or unformatted file? [f/u]:
```

PTINGRID next performs its function of locating points with respect to model grid cells. The example below shows part of its output file.

```
point_id    easting        northing      row      column      array_value
40050,      432757.0010,   7251364.668,  47,      22,         7.54648018
40051,      432532.6500,   7251332.776,  47,      21,         7.64460993
40056,      431938.7510,   7252252.603,  --,      --,         --
40057,      432189.9940,   7252530.667,  44,      22,         8.02033043
40063,      431794.5960,   7253020.996,  43,      22,         9.02902985
40064,      431852.4520,   7252682.867,  44,      22,         8.02033043
40065,      431992.0970,   7252806.516,  44,      22,         8.02033043
40071,      430844.4660,   7252185.675,  43,      20,         9.43745041
40078,      431043.5090,   7251663.817,  44,      19,         12.4602003
40080,      432970.5330,   7253641.892,  44,      25,         5.37747002
40082,      432885.5060,   7253795.232,  43,      25,         5.65251017
40083,      430646.7010,   7252430.711,  43,      20,         9.43745041
40084,      430700.2510,   7252953.946,  42,      20,         10.6384001
40085,      431519.9990,   7251789.264,  45,      20,         10.8271999
40091,      432406.3710,   7254192.785,  42,      25,         6.50939989
40093,      432185.1520,   7253514.981,  43,      23,         8.67656994
40094,      432018.3670,   7253175.856,  43,      23,         8.67656994
40095,      432184.3950,   7253668.732,  43,      24,         5.87722015
40204,      427669.2340,   7258198.290,  30,      22,         20.6464005
40205,      427556.2820,   7258320.658,  30,      22,         20.6464005
40210,      426825.9000,   7258378.399,  29,      21,         17.2196007
40221,      426378.6310,   7258006.921,  29,      20,         21.3871994
```

**Extract from a PTINGRID output file.**

The PTINGRD output file resembles a bore information file in that the first column contains bore identifiers (as read from the bore listing file) and subsequent columns contain information pertaining to the bores cited in the first column. However column headers are added to assist the user in identifying the contents of each column. Note that if a point does not lie within the bounds of the finite-difference grid its row and column numbers are given as "--", as is the array value of the cell in which it lies.

If a real or integer array was not read by PTINGRID the final column of its output file is omitted.

**Uses of PTINGRID**

It is often necessary to know the cell in which a particular point lies. This can be useful, for example, in assisting with manual data entry of pumping rates prior to a MODFLOW run. Alternatively, PTINGRID can be used in conjunction with program PT2ARRAY to evaluate the total amount of pumping taking place from each cell,

where certain cells may hold multiple bores. Thus PT2ARRAY can be used to construct a "pumping array" for the use of MODFLOW graphical user interfaces such as PMWIN which allow such data to be supplied in array format. The same bore coordinates file and bore listing file used by PT2ARRAY in building the pumping array can then be used by PTINGRID to establish the total pumping rate from any cell containing one or more bores.

If the bore listing and bore coordinates files supplied to PTINGRID tabulate active cell centres (as obtained with the help of program GRID2PT), "pumping cells" can be identified and plotted on a map of the study area using the cell centre coordinates reproduced in the second and third columns of a PTINGRID output file. Total cell pumping rates, as tabulated in the sixth column, can be plotted as proportional symbols if desired.

**See Also**

See also GRID2PT, PT2ARRAY.

# QDIG2DXF

**Function of QDIG2DXF**

Program QDIG2DXF rewrites the files produced by the shareware digitizing program QDIGIT as DXF files. This allows for the ready importation of spatial data digitized using QDIGIT into most visualisation, GIS, mapping and display software, as well as most graphical model preprocessors. As well as providing the translation functionality necessary for DXF file generation, QDIG2DXF is also able to thin over-sampled line data generated by QDIGIT.

**Using QDIG2DXF**

QDIG2DXF commences execution with the prompt:

```
 Enter name of QDIGIT-generated file:
```

Enter an appropriate filename. Next QDIGIT inquires as to what kind of file this is:

```
 Is this file a line, points or well file? [l/p/w]:
```

The format for each of these QDIGIT-generated files is discussed in Sections 2.13 to 2.16 of Part A of this manual. These file types refer to the files produced using the QDIGIT digitizing options of the same name. Note that the file produced by the QDIGIT "contour" option can be classed as a "line file" as far as the process of DXF file generation is concerned.

If you indicate that a file is of the QDIGIT "line file" type, QDIG2DXF next prompts:

```
 Enter trimming distance:
```

When QDIGIT is used with some digitizers, point coordinates are recorded in numbers too great to be useful. The resulting output files are long and cumbersome with many coordinates repeated; see below for a further discussion. QDIG2DXF provides the means to eliminate many of the points along such over-sampled lines. By entering a distance (in map units) in response to the above prompt, QDIG2DXF will ensure that culling of excess data points is carried out such that inter-point distances are roughly of this magnitude. In order to save on processing time, QDIGIT does not rigorously ensure that the point density is exactly equal to the trimming distance entered here. It simply culls points using this distance as a rough guide.

If the QDIGIT file to be processed by QDIG2DXF is not a line file but a points or well file, QDIG2DXF does not ask for a trimming distance. Instead it prompts:

```
 Represent points by a cross of what length?
```

Enter an appropriate distance (in map units); thus if you enter "100" and the map was digitized in meters, QDIG2DXF will generate two intersecting lines for each point comprising a cross of length 100m.

QDIG2DXF next prompts for the name of the file to which it should write the geographical information in DXF format:

```
 Enter name for output DXF file:
```

Enter an appropriate filename; an extension of "DXF" is recommended.

**Some Hints on using QDIGIT**

QDIGIT has been designed to work with as many different types of digitizer as possible. However when used with some digitizers, its operation is a little awkward when digitizing lines and contours.

When digitizing a line the user is instructed to hold down the mouse and trace along the line. Sometimes, with the button held down in this way, an overwhelming number of points is recorded on the QDIGIT output file. Furthermore, once the button has been pressed, points continue to be generated even after the button has been subsequently raised; data generation ceases only when the <Enter> key is pressed. The trimming functionality of QDIG2DXF culls a line file produced by QDIGIT in this fashion, removing extraneous data points before recording the line data in DXF format. See Section 3 of Part A of this manual for more details.

When digitizing a line comprised of a number of straight segments it is unwise to trace the line with the digitizer button held down; the latter method is appropriate for natural features such as streams and contour lines. For roads, property boundaries etc, the user should click on each boundary corner. While QDIGIT may still generate a multitude of points until the <Enter> key is pressed, all points will be duplicates of the latest corner at which the digitizer button was pressed. Once again, QDIG2DXF's trimming functionality will cull the sequence of point coordinates recorded on its line file such that only a single incidence of the coordinates of each corner remain.

**Uses of QDIG2DXF**

Preparation of a base map is a necessary first step in model pre-processing. In many cases a modeller's client will be able to supply him/her with one or more DXF (or other) files of the model area. In other cases this information will not be available. In this case, the use of QDIGIT together with QDIG2DXF allows a user to generate DXF files of a study area cheaply and without difficulty.

It should be noted that map information digitized with the help of QDIGIT has many uses. Translation to DXF format for base map production is one such use. However points and well files can be used for data posting within most mapping and display packages. QDIGIT contour files can be used for the production of digital elevation models as well as model initial condition arrays.

**See Also**

See also QDIG2XYZ.

# QDIG2XYZ

**Function of QDIG2XYZ**

QDIG2XYZ reformats the data contained in a QDIGIT-generated contour file as a "*xyz*" data file. In the process of making this translation it reduces data density according to user specifications.

**Using QDIG2XYZ**

QDIG2XYZ commences execution with the prompt:

```
 Enter name of QDIGIT-generated contour file:
```

in response to which a suitable filename should be supplied. Next QDIG2XYZ prompts:

```
 Enter trimming distance:
```

Enter a distance in map units. QDIG2DXF uses the trimming distance to reduce contour line data density such that points along any line within the *xyz* data file which it generates are between half the trimming distance and the full trimming distance apart; superfluous points are culled to achieve this outcome. QDIG2XYZ's ability to carry out trimming of its QDIGIT-supplied input data overcomes the problem of oversampling that can occur when QDIGIT is used with certain digitizers.

Next QDIG2XYZ asks:

```
 Enter name of output "xyz" file:
```

Supply an appropriate filename. Once you press the <Enter> key, QDIG2XYZ reads the QDIGIT-generated input file, generating an *xyz* output file in accordance with the trimming specifications supplied above. The following example illustrates a QDIG2XYZ output file.

```
402071.300          7258340.480          330.000
402146.700          7258343.580          330.000
402219.630          7258354.220          330.000
402295.040          7258357.320          330.000
402367.900          7258352.850          330.000
402443.350          7258360.990          330.000
402518.800          7258366.610          330.000
402591.750          7258374.730          330.000
402664.710          7258382.860          330.000
402740.180          7258388.480          330.000
402815.680          7258399.150          320.000
402896.220          7258409.850          320.000
402974.240          7258418.020          320.000
403047.250          7258428.670          320.000
403120.270          7258439.320          320.000
403203.370          7258452.570          320.000
403281.480          7258473.350          320.000
403357.040          7258484.020          320.000
```

**Extract from a QDIG2XYZ output file.**

As is apparent from the above illustration, the *xyz* file produced by QDIG2XYZ consists of three columns of data, the first two of which contain eastings and northings of points along a contour line. The final column contains the contour line elevation.

**Uses of QDIG2DXF**

An *xyz* data file produced by QDIG2XYZ can be supplied to just about any package which is capable of undertaking spatial interpolation for the purpose of digital elevation model and/or grid generation. Hence digitized contours can be used to construct model elevation and initial condition arrays.

**See Also**

See also QDIG2DXF.

# REAL2INT

**Function of REAL2INT**

Program REAL2INT builds a MODFLOW/MT3D-compatible integer array based on the data contained in a MODFLOW/MT3D-compatible real array. Integer array elements are assigned on the basis of either real array values or real array value ranges.

**Using REAL2INT**

A settings file `settings.fig` must be present in the directory from which REAL2INT is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

REAL2INT begins execution with the prompt:

```
 Enter name of grid specification file:
```

to which you should respond with the appropriate filename. In the event that REAL2INT is able to read a default grid specification filename from a filename file (`files.fig`) located in the current directory, that filename will be included in the above prompt. If so, accept it by pressing <Enter> or type in the more appropriate filename.

Next REAL2INT requests the name of a file holding a model-compatible real array:

```
 Enter name of real array file:
```

When responding to this prompt keep in mind the real array file naming convention set out in Section 2.17 of Part A of this manual.

Next REAL2INT asks the user how it should construct an integer array based on the real array that it has just read:

```
 Use ranges or individual values to build integer array?  [r/i]:
```

If "i" is selected, REAL2INT constructs the integer array by matching integers to real numbers on a one-to-one basis. It peruses the real array, establishing how many different real numbers are represented in it, and then presents each number to the user (in ascending order), asking him/her for the corresponding integer to write to the integer array:

```
 The following numbers have been detected in the real array:-
 Enter corresponding integers.
   enter integer corresponding to real number 0.340:
   enter integer corresponding to real number 1.995:
   enter integer corresponding to real number 2.345:
             etc.
```

Note that there is an upper limit (set at 100) on how many different real numbers can occur within the real array if one-to-one real-to-integer correspondence is used for integer array construction; this limit can be raised if desired through making an alteration to the REAL2INT source code and recompiling it. Note also that a user can supply the same integer to correspond to different real numbers occurring in the real array, thus reducing the number of zones occurring in the integer array from that found in the real array. REAL2INT prompts for integers in the above fashion until an integer has been supplied for every real number occurring in the real array.

Alternatively, if instead of one-to-one real-to-integer correspondence, you had informed REAL2INT that the integer array should be constructed on the basis of value ranges found in the real array, REAL2INT prompts:

```
Enter the range boundaries:-
Use "+i" for "plus infinity" to terminate.
  range number 1:   minus infinity to: -4.345
  range number 2:          -4.345   to:
              etc.
```

If the range method is used for integer array construction, REAL2INT imposes no upper limit on the number of different numbers occurring within the real array. A user defines value ranges in response to the above prompts, signalling "+i" as the top of the highest range. An integer corresponding to each of these ranges must then be supplied:

```
Now enter the integers corresponding to these ranges:-
  enter integer for range minus infinity to -4.345:
  enter integer for range   -4.345       to -2.447:
              etc.
```

Once all integers have been supplied for either the one-to-one or range methods, REAL2INT prompts for the name of the file to which it must write the integer array that it builds:

```
 Enter name for output integer array file:
```

to which you should reply with an appropriate filename, remembering the integer array naming conventions outlined in Section 2.10 of Part A of this manual.

After writing the integer array REAL2INT terminates execution.

**Uses of REAL2INT**

REAL2INT is used for the construction of integer arrays expressing model grid zonation. Such integer zonation arrays can then be used for the construction of other real arrays as they are needed at subsequent stages of model pre- and/or postprocessing, (for example in parameter assignment using program INT2REAL). Also, using programs ZONE2DXF and ZONE2BLN, a DXF or SURFER blanking file can be built from an integer array derived from a model real array, thus facilitating the display of model real array zonation within a GIS or mapping package.

**See Also**

See also INT2REAL.

# REAL2MIF

**Function of REAL2MIF**

REAL2MIF generates a MAPINFO MIF/MID file pair containing the geographical and cell information pertaining to a MODFLOW/MT3D-compatible real array. The files generated by REAL2MIF can be used by MAPINFO (and other geographical information systems) to import a model real array for GIS-based model pre/postprocessing and display.

**Using REAL2MIF**

A settings file `settings.fig` must be present in the directory from which REAL2MIF is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution REAL2MIF prompts:

```
 Enter name of grid specification file:
```

Type in the name of the grid specification file pertinent to the current model. Note that if a filename file (`files.fig`) resides in the directory from which REAL2MIF is invoked, its name will appear with the above prompt as the default grid specification filename; press <Enter> to accept the default or type in an alternative filename.

REAL2MIF reads a real array and an integer array. First it asks for the name of the file holding the real array. This is the array whose contents are to be written to the MID file for subsequent uploading into a GIS. REAL2MIF prompts:

```
 Enter name of real array file:
```

to which you should respond with an appropriate filename noting the convention for real array filename extensions used by the Groundwater Data Utilities; see Section 2.17 of Part A of this manual. Next REAL2MIF prompts:

```
 Enter name of window integer array file:
```

The window integer array acts as a mask or "cookie-cutter" over the real array. Only cells within the latter array whose corresponding cells within the former array are non-zero appear in the MIF/MID files written by REAL2MIF.

REAL2MIF next requests the names of the files to which it should write the geographical and array information respectively pertaining to the "active window" (as defined by the window integer array) of the real array; note that array geographical information is taken from the grid specification file read at the beginning of REAL2MIF's execution. REAL2MIF prompts:

```
 Enter name for output "MIF" file:
```

```
 Enter name for output "MID" file:
```

Then it asks:

```
 Enter AMG zone number of model area  [47 - 58]:
```

in response to which the Australian Map Grid zone number of the current study area should be entered. (If you do not live in Australia provide any zone number within the range indicated. After REAL2MIF has finished execution you should replace the third line of the MIF file generated by REAL2MIF with more appropriate `Coordsys` information from file `MAPINFOW.PRJ` supplied with MAPINFO.)

REAL2MIF then calculates the coordinates of the corners of each "active" cell of the finite-difference grid and transfers this data, in appropriate format, to the MIF file whose name was supplied above. The values of "active" real array elements are written to the user-nominated MID file.

**Uses of REAL2MIF**

REAL2MIF provides the means whereby real array data can be imported into a geographical information system (GIS). Once imported into the GIS, cells within the grid can be displayed as "regions" or "polygons", each with its own attributes. Three attributes are uploaded through the MID file generated by REAL2MIF, these being the row and column numbers and the array value for each cell. The latter can be displayed (for example as a thematic map) in conjunction with other mapping information covering the study area. Within the GIS cell real array values (not row and column numbers) can be edited with reference to any pertinent geographical information covering all or part of the study area. The data content of the modified real array can then be downloaded as a real array table, and using program TAB2REAL, rewritten in proper array format. Employed in this manner, the conjunctive use of REAL2MIF and TAB2REAL allow a GIS to be used as a model pre- and postprocessor.

**See Also**

See also INT2MIF, TAB2INT, TAB2REAL.

# REAL2SRF

**Function of REAL2SRF**

REAL2SRF rewrites a MODFLOW/MT3D-compatible real array as a SURFER grid file. Thus model results can be directly contoured (or displayed as a surface) without any intermediate gridding.

**Using REAL2SRF**

A settings file `settings.fig` must be present in the directory from which REAL2SRF is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

Upon commencement of execution REAL2SRF prompts:

```
 Enter name of grid specification file:
```

If a filename file (`files.fig`) is present in the directory from which REAL2SRF is invoked, its name will appear with the above prompt. In this case press <Enter> to accept the default or type in the correct filename. REAL2SRF needs to read the grid specification file in order to obtain the dimensions and measurements of the current model grid. If the grid is not uniform (ie. the elements of the MODFLOW *delr* vector are not all equal and the elements of the MODFLOW *delc* vector are not all equal), REAL2SRF terminates execution with an appropriate error message, for this is a necessary precondition for representing a model grid as a SURFER grid.

Next REAL2SRF prompts for the name of a MODFLOW/MT3D-compatible real array file:

```
 Enter name of real array file:
```

to which you should respond by supplying an appropriate filename, taking note of the real array filename conventions discussed in Section 2.17 of Part A of this manual. Unless a filename with an extension of "REF" or "REU" is supplied, REAL2SRF inquires as to whether the file is formatted or unformatted:

```
 Is this a formatted or unformatted file?  [f/u]
```

Enter "f" or "u" as appropriate.

In model-generated real arrays, certain (normally high) values can represent certain conditions. For example a head value of 999.99 may be used to fill inactive cells, and a value of 1.0E30 may represent a cell that has gone dry. Such values should not be contoured by SURFER; they should be blanked instead. SURFER employs the value 1.70141E38 to demarcate a blanked grid element.

In order that it may be aware of which cells should be blanked in the SURFER grid file which it produces, REAL2SRF prompts:

```
 Enter blanking threshold value for this array:
```

Any real array elements whose absolute value exceeds the number which you enter here will be assigned the value of 1.70141E38 by REAL2SRF; hence they will not be contoured.

Finally SRF2REAL prompts for the name of the SURFER grid file which it should write:

```
 Enter name for SURFER grid file:
```

Supply an appropriate filename; an extension of "GRD" is highly recommended. REAL2SRF then rewrites the model real array as an ASCII SURFER grid file and terminates execution. The grid file so written can be used by SURFER through selection of its "Map/Contour" and "Map/Surface" menu items.


**Uses of REAL2SRF**

If a model grid is uniform, importation of model-generated real arrays into SURFER provides an ideal mechanism for the production of presentation-quality results. The fact that a model array can be reformatted as a SURFER grid with a one-to-one relationship between model cells and SURFER grid points allows SURFER to contour the model results directly, without any intermediate re-gridding (as required by many other model-to-SURFER translators). A contour map or surface produced by SURFER on the basis of the translated model real array can then be shaded, overlain on maps, annotated to produce an informative pictorial representation of model predictions, etc.

If the model grid is oriented such that its row direction is east-west, REAL2SRF retains the map coordinate system used by the model when it writes the SURFER grid file (grid maximum and minimum coordinates are written to the third and fourth lines of the SURFER grid file). Hence maps of the study area can be directly overlain on the contour map or surface generated by SURFER. However if the row direction of a finite difference grid does not point east, REAL2SRF rotates the finite difference grid before conversion to SURFER format, this being required because a SURFER grid must be orientated with its *x*-direction pointing east and its *y*-direction pointing north. Rotation takes place about the top left corner of the grid until the grid row direction is oriented easterly and the grid column direction is oriented northerly. The grid is also displaced such that the top left corner of the model grid has (*x,y*) coordinates of (0,0). In order that map and point data can be overlain on contour maps and surfaces produced by SURFER on the basis of the displaced and rotated grid file, relevant DXF, blanking and point data files must also be rotated and displaced; this can be accomplished using programs ROTDXF, ROTDAT and ROTBLN documented in this manual. Note that such rotated maps can be re-oriented for display in SURFER version 6 and later using the map rotation facility. You should specify a rotation angle equal to that found on the second line of the grid specification file, ie. the angle between east and the grid row direction; see Fig. 2.1 of Part A of this manual.

**See Also**

See also ROTBLN, ROTDAT, ROTDXF and SRF2REAL.

# REAL2TAB

**Function of REAL2TAB**

REAL2TAB writes a MODFLOW/MT3D-compatible real array in three-column real array table format. As such it performs the inverse operation to program TAB2REAL. It can be useful as a MODFLOW/MT3D post-processor in a composite model run by PEST, where MODFLOW/MT3D outputs at certain cell centres are matched to the values contained in an "observation array" through the inversion process.

**Using REAL2TAB**

As for other members of the Groundwater Data Utility suite, a settings file `settings.fig` must be present in the directory from which REAL2TAB is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted real and integer array files.

As it commences execution REAL2TAB prompts:

```
 Enter name of grid specification file:
```

Respond to this prompt with an appropriate filename. Alternatively, if REAL2TAB has read the name of a grid specification file from a filename file (`files.fig`) resident in the current directory, that grid specification filename will appear with the above prompt. Press <Enter> to accept the default or type in the correct filename.

Note that, in responding to this prompt, or to any other prompt issued by any of the members of the Groundwater Data utilities, an "e" signifies a request to return to the previous prompt. Through this mechanism, the user can "rewind" execution of the program.

Next REAL2TAB prompts:

```
 Enter name of real array file:
```

Enter the name of a formatted or unformatted file holding a MODFLOW/MT3D-compatible real array. As is the protocol for members of the Groundwater Data Utilities, if the provided filename has an extension of ".ref" REAL2TAB assumes that the file is formatted. However if it possesses an extension of ".reu", the file is assumed to be unformatted. If the extension is neither of these, REAL2TAB prompts for the formatted/unformatted status of the file.

Next REAL2TAB prompts:-

```
 Enter name of window integer array file:
```

REAL2TAB uses the contents of this file to determine which real array elements are to be represented in its output file; any cell for which the corresponding integer array element is zero will be omitted. The "window integer array" file can be formatted or

unformatted. As is the protocol for members of the Groundwater Data Utilities, if the provided filename has an extension of ".inf" REAL2TAB assumes that the file is formatted. However if it possesses an extension of ".inu", the file is assumed to be unformatted. If the extension is neither of these, REAL2TAB prompts for the formatted/unformatted status of the file.

REAL2TAB's next prompt is:-

 Enter name for output real array table file:

Enter a filename; there is no protocol pertaining to the extension used by files of this type, so any extension will do. Then REAL2TAB asks:

 Write native or log10 values to output file?  [n/l]:

Enter "n" or "l" as appropriate. If the "l" option is chosen, and if there are any zero or negative real array values within the "active region" of the grid (as defined by non-zero integer array values), REAL2TAB will report this and terminate execution.

The figure below shows part of a real array table file generated by REAL2TAB. Each line of this file corresponds to a model cell (with cells pertaining to window integer array elements of zero omitted). The first two items on each line are the cell row and column numbers. Then follows the corresponding real array value. Note that real array element vales are recorded with maximum numerical precision. Thus if REAL2TAB is used as part of a composite model in a PEST inversion process, derivatives with respect to model outputs calculated through finite differences will be reasonably accurate.

```
  36       2      4.733733E-01
  36       3      4.694479E-01
  36       4      4.653012E-01
  36       5      4.609686E-01
  36       6      4.565133E-01
  36       7      4.520330E-01
  36       8      4.476580E-01
  36       9      4.435370E-01
  36      10      4.398057E-01
  36      11      4.365453E-01
  36      12      4.337443E-01
  36      13      4.312836E-01
  36      14      4.289554E-01
  36      15      4.265116E-01
  36      16      4.237257E-01
```

**Part of a real array table file written by REAL2TAB.**


**Uses of REAL2TAB**

REAL2TAB can be used to transfer real array data to a GIS or spreadsheet. It can also be used to re-write real array data in a format in which it is easily read using a PEST instruction set. This is useful if the contents of a model-generated real array are to be compared with those of an "observation real array" through the inversion process.

**See Also**

See also REAL2MIF, TAB2INT, TAB2REAL.

# REPARRAY

**Function of REPARRAY**

REPARRAY performs an almost identical role to that of program ARRAYREP from the PEST MODFLOW/MT3D Utilities, ie. it replaces a data array in an existing MODFLOW or MT3D input file by an external array, generating a new MODFLOW or MT3D input file in the process. Thus a MODFLOW or MT3D-compatible real array can be generated by software other than the user's graphical MODFLOW interface and "pasted" into the MODFLOW or MT3D dataset generated by that graphical user interface prior to running either of these models.

**Using REPARRAY**

A settings file `settings.fig` must be present in the directory from which REPARRAY is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

Upon commencement of execution REPARRAY, like most of the programs of the Groundwater Data Utilities, prompts for the name of the grid specification file for the current case. If this file is cited in a filename file (ie. `files.fig`) residing in the directory from which REPARRAY is run, that filename is presented with the above prompt. As usual, the user can either accept the default by pressing the <Enter> key, or supply the name of an alternative grid specification file.

Next REPARRAY prompts:-

```
 Enter name of real array file:
```

See Section 2.17 of part A of this manual for the specifications of a real array file. Enter an appropriate filename in response to the above prompt.

Next REPARRAY prompts for the name of an existing MODFLOW or MT3D input file in which an included array must be replaced by the array residing in the real array file whose name was supplied in response to the previous prompt:-

```
 Enter name of MODFLOW/MT3D input file:
```

REPARRAY needs to know if this is a MODFLOW or MT3D input file, so it prompts:-

```
 Is this a MODFLOW or MT3D input file  [f/t]?
```

Respond with "`f`" or "`t`" as appropriate. Note that, unlike program ARRAYREP from the PEST MODFLOW/MT3D Utilities, REPARRAY can work with input files for both the MODFLOW88 and MODFLOW96 (and later) versions of MODFLOW, as well as with MT3D input files.

Two methods are used to locate the array residing within the MODFLOW or MT3D input file which is targeted for replacement. In each case it is the line preceding the array, rather than the array itself, which is located. This header line informs MODFLOW or MT3D how the array is to be read. Most commercial MODFLOW preprocessors add some text to this line (not read by MODFLOW) in order to facilitate user identification of the following array. The text may be, for example, "`transmissivity - layer 1`". REPARRAY can either use this text to find the pertinent array header, or it can count lines from the top of the file to the array header. REPARRAY asks the user for the most appropriate method:-

```
Locate array using text of line numbers  [t/l]?
```

If text is chosen, the user is next prompted for an appropriate text string:-

```
Enter text:-
```

Supply enough text to uniquely locate the array header on the MODFLOW or MT3D input file (you need only supply a substring of the text actually present). Note, however, that if the text does not uniquely demarcate a header (for example if the user supplied the string "`trans`" instead of "`transmissivity - layer 1`" in response to the above prompt, then every array which follows a header located using the text string will be replaced by the array in the user-supplied real array file (REPARRAY will inform you of this). Note also that the text search is case-insensitive.

Alternatively, if the users specifies that the appropriate array header be located using a line number, REPARRAY prompts:-

```
Enter line number:
```

in response to which an appropriate line number (pertaining to the array header rather than to the array itself) must be supplied.

Finally REPARRAY prompts:-

```
Enter name for new MODFLOW/MT3D input file:-
```

Upon receipt of this filename, REPARRAY reads the pre-existing MODFLOW or MT3D input file, transferring its contents to the new MODFLOW or MT3D input file, making no alterations except for the appropriate array replacement. Note that the replaced array may be written to the new model input file with a different format from that with which it was recorded in the original input file. However this does not matter, as MODFLOW or MT3D is informed of the format through the array header written to the new input file.

## Uses of REPARRAY

REPARRAY is of great use where a model input array is constructed using software other than the user's commercial MODFLOW/MT3D graphical user interface. For example this array may have been constructed using one or more of the Groundwater

Data Utilities. REPARRAY is especially useful where array construction takes place within a "composite model" encapsulated within a batch file, and where this composite model is run by PEST. Thus PEST can be used to estimate parameters governing construction of the array (for example spatial interpolation parameters, zone properties, etc). After the array-building software is run from within the batch file, REPARRAY is run to "paste" the array into an existing MODFLOW dataset prepared with the help of a commercial MODFLOW/MT3D graphical user interface. Thus the user can combine the use of the graphical user interface and with that of independent, special-purpose, array-generating software together with MODFLOW in a composite model for which parameters are estimated using PEST.

**See Also**

See also INT2REAL, TWOARRAY.

# ROTBLN

**Function of ROTBLN**

ROTBLN rotates all line data represented in a SURFER blanking file about the top left corner of the finite-difference grid and positions the latter at coordinates (0,0). In this way SURFER can be used to superimpose line data contained in blanking files on contour maps, the latter being based on SURFER grid files generated by program REAL2SRF for a finite difference grid of uniform dimensions but for which the row direction is not oriented east-west. Blanking files rotated by ROTBLN can also be used for partial blanking of a REAL2SRF-generated grid file.

**Using ROTBLN**

Like many of the Groundwater Data Utilities, ROTBLN commences execution with the prompt:

```
 Enter name of grid specification file:
```

Supply the appropriate filename. Alternatively, if a filename file `files.fig` is present in the directory from which ROTBLN is run, a default filename read from the filename file may appear in the above prompt. In this case either type a more appropriate filename or press <Enter> to accept the default.

ROTBLN next requests the name of the blanking file which it must rotate, and a name for the blanking file to which it should write the rotated geographical information. An extension of "`BLN`" is the SURFER convention for files of this type.

```
 Enter name of SURFER blanking input file:
 Enter name for rotated SURFER blanking output file:
```

Supply appropriate filenames. ROTBLN then reads the blanking input file, writing the rotated blanking output file as it goes.

**Uses of ROTBLN**

As is discussed in the documentation of program REAL2SRF, SURFER can be used to directly contour MODFLOW and MT3D-generated head, drawdown, concentration and other arrays without an intermediate gridding step if the finite-difference grid is uniform and can thus be emulated by a SURFER grid (the latter forming the basis of SURFER's contouring and surface-generation functionality). However the process becomes a little complicated if the model grid is not oriented with its row direction east-west, for a SURFER grid must always be oriented with its *x* direction pointing east and its *y* direction pointing north. In this case REAL2SRF actually rotates the finite difference grid (and hence the geographical relevance of information contained in arrays pertaining to the grid) such that it is oriented with its row direction east-west and its column direction north-south. It also places the top left corner of the grid at (0,0).

A grid file produced by REAL2SRF is immediately readable by SURFER for contouring and surface-generation purposes. However if geographical information pertaining to the study area is to be overlain on the contour map, this too must be rotated and displaced in the same fashion as the contoured array data. The propose of ROTBLN is to effect this translation where geographical line information is contained within SURFER "blanking" files. (Note that no such rotation and translation is necessary if the finite-difference grid is oriented with its row direction east-west, for REAL2SRF then produces a SURFER grid file based on real-world coordinates.)

Thus to use SURFER to contour model results for a finite-difference model in which the grid is uniform but not oriented with its row direction east-west, and to superimpose map data residing in SURFER blanking files on this contour map, the following steps should be followed:

- Use REAL2SRF to generate a SURFER grid file based on a two-dimensional model results array.
- Use ROTBLN to rotate any blanking files containing map data that you wish to superimpose on the contour map.
- Import the grid and blanking files into SURFER and superimpose them.
- Use the SURFER "Arrange/Rotate" function to rotate the resulting map into its correct orientation. (Note however that the SURFER-generated axes should be removed from the resulting diagram as the labels pertain to a rotated, translated coordinate system that has no real-world significance.)

The contents of a rotated and displaced SURFER blanking file produced by program ROTBLN can also be used for partial blanking of a rotated and displaced grid file produced by program REAL2SRF; in this manner contours can be displayed only over those parts of a study area judged appropriate for a particular application.

**See Also**

See also REAL2SRF, ROTDAT, ROTDXF.

# ROTDAT

**Function of ROTDAT**

When a finite difference grid is uniform, SURFER can be used to display model results and generate initial condition arrays by allowing the SURFER grid to emulate the model grid. However where the latter is not oriented with its row direction east-west, a model grid, together with any geographical data pertaining to the model, must be rotated until its row direction is oriented east-west and can thus coincide with a SURFER grid. ROTDAT carries out this rotation and translation for model-related data residing in files comprised of data columns, one of which contains point east coordinates and another of which contains point north coordinates.

**Using ROTDAT**

Like many of the Groundwater Data Utilities, ROTDAT commences execution with the prompt:

```
 Enter name of grid specification file:
```

Supply the appropriate filename. Alternatively, if a filename file `files.fig` is present in the directory from which ROTDAT is run, a default filename read from the filename file may appear in the above prompt. In this case either type in a more appropriate filename or press <Enter> to accept the default.

Next ROTDAT prompts:

```
 Enter name of input data file:
```

Here provide the name of a file in which data is structured in columns (normally a "bore information file" - see Section 2.4 of Part A of this manual). One of these columns must contain east coordinates and another must contain corresponding north coordinates. The role of ROTDAT is to rewrite these columns with rotated and displaced east and north coordinates while leaving all other columns unchanged. So it prompts:

```
 In which column are the east coordinates?
 In which column are the north coordinates?
```

to which you should respond with the appropriate column numbers. ROTDAT then asks for the name of the file to which it should write its output data (viz. a file containing all the data contained in the original file but with appropriately transformed east and north coordinates):

```
 Enter name for rotated output data file:
```

in response to which an appropriate filename should be supplied.

ROTDAT then reads the input file, calculating new point coordinates and writing these new coordinates, together with all other data from the input file, to the user-specified output file.

**Uses of ROTDAT**

ROTDAT can be used in two important aspects of model pre- and postprocessing. The first aspect is in the construction of initial conditions arrays (initial heads in the case of MODFLOW, initial concentrations in the case of MT3D). The second is in the display of map data superimposed on a contour map generated by SURFER on the basis of a REAL2SRF-constructed SURFER grid file.

*Initial Conditions Array*

The following discussion will focus on the construction of a MODFLOW initial heads array; however it could equally well apply to the construction of a MT3D initial concentration array, for the principles are identical.

If a model finite-difference grid is uniform, SURFER can be used to construct an initial heads array by interpolating from a set of randomly-disposed data points (eg. boreholes) onto a grid that emulates the model finite difference grid. The SURFER grid can then be translated to MODFLOW-compatible form using program SRF2REAL. Where the model grid is oriented with its row direction east-west, this is a straightforward procedure; a user must simply ensure that the nodes of the SURFER grid coincide with the cell centres of the finite-difference grid. However where a model grid is oriented in a manner such that its row direction deviates from east-west, the grid must be rotated until this deviation is zero; so, too, must any data associated with the grid, including data used in the construction of an initial heads array over the grid. If ROTDAT is used to perform the necessary data rotation, then a real array built as a SURFER grid file will automatically have undergone the correct amount of rotation. However as a real array contains no geographical information, it is also immediately useable by the original, unrotated model.

So to construct an initial heads array for a uniform grid whose row direction is not oriented east-west, first use ROTDAT to rotate all data about the top left corner of the finite-difference grid; ROTDAT also translates the grid top left corner to (0,0), and in so doing translates the data an equivalent amount. Then, using SURFER, interpolate the data to a SURFER grid which has the same number of lines in the *x* and *y* directions as there are columns and rows respectively in the finite difference grid. The grid must be placed within the new, rotated and displaced coordinate system, such that its minimum line *x*-coordinate coincides with the first column of model grid cell centres; this is one half row width (ie. *delr/2.0*) higher than (0,0), the new location of the upper left corner of the finite difference grid. Similarly the maximum line *y* coordinate must coincide with the upper row of cell centres; this is one half column width (ie. *delc/2.0*) below (0,0), the new location of the upper left corner of the finite-difference grid. The spacing of lines in the *x* direction must equal *delr*, ie. the (uniform) model row-direction cell width; the spacing of lines in the *y* direction must equal *delc*, ie. the (uniform) model column-direction cell width. SURFER should be instructed to store the grid in an ASCII file for subsequent conversion to a

MODFLOW/MT3D-compatible real array by program SRF2REAL. Such an array is ready for immediate use as a model initial conditions array.

Note that if the finite difference grid is already oriented with its row direction east-west, ROTDAT is not required for data rotation. Direct superimposition of SURFER grid nodes and model grid cell centres is still essential. However the SURFER grid can be constructed using real world coordinates.

*Posting Data on SURFER-Generated Model Results Contours*

The discussion of programs REAL2SRF and ROTDXF describes the manner in which SURFER can be used to contour model results without the intervention of any interpolation steps; as in the generation of initial conditions arrays, SURFER can only be used in this capacity if the finite-difference grid is uniform. If the grid is uniform but not oriented with its row-direction east-west, the grid must be rotated until its row direction is thus oriented in order to conform with SURFER's orientation requirements. Model arrays are rotated in this manner using program REAL2SRF; rotation takes place about the top left corner of the finite difference grid and the latter is translated to (0,0).

If real-world data is to be posted on a contour map produced in this way, it must be similarly rotated and translated. If such data is housed in a file comprised of columns, two of which contain point eastings and northings, the necessary rotation and translation can be accomplished using program ROTDAT. Thus, for example, borehole locations can be posted on a contour map of model results. If the former are labelled with, for example, water elevations measured in these bores (presumably this information comprising one of the columns of the rotated data file), a comparison can be made between contoured water levels as generated by the model and measured heads at field observation points.

**See Also**

See also REAL2SRF, ROTBLN, ROTDXF, SRF2REAL.

# ROTDXF

**Function of ROTDXF**

ROTDXF rotates all geographical elements in a DXF file about the top left corner of the finite-difference grid and positions the latter at (0,0). In this way geographical data can be superimposed on contoured model results where the contours are generated directly from a model real array by a package for which gridded data must be disposed with the grid row direction pointing east, but where the model grid is oriented in an arbitrary direction with respect to east. It thus complements program REAL2SRF which translates a real array based on a uniform grid into SURFER grid format.

**Using ROTDXF**

Like many of the Groundwater Data Utilities, ROTDXF commences execution with the prompt:

```
 Enter name of grid specification file:
```

Supply the appropriate filename. Alternatively, if a filename file (`files.fig`) is present in the directory from which ROTDXF is run, a default filename read from the filename file may appear in the above prompt. In this case either type in a more appropriate filename or press <Enter> to accept the default.

ROTDXF next requests the name of the DXF file which it must rotate, and a name for the file to which it should write the rotated geographical information.

```
 Enter name of DXF input file:
 Enter name for rotated DXF output file:
```

Supply appropriate filenames. ROTDXF then reads the DXF input file, writing the rotated output file as it goes.

**Uses of ROTDXF**

As is discussed in the documentation to program REAL2SRF, a contouring program such as SURFER can be used to directly contour MODFLOW and MT3D-generated head, drawdown, concentration and other arrays without an intermediate gridding step if the finite-difference grid is uniform and can thus be emulated by a SURFER grid. However the process becomes a little complicated if the model grid is not oriented with its row direction east-west, for a SURFER grid must always be oriented in this direction. In this case REAL2SRF actually rotates the finite difference grid (and hence the geographical relevance of information contained in arrays pertaining to the grid) such that the latter is oriented with its positive row direction oriented easterly and its column direction oriented north-south. It also places the top left corner of the grid at (0,0) in this new, rotated, coordinate system.

A grid file produced by REAL2SRF is immediately readable by SURFER for contouring and surface-generation purposes. However if geographical information pertaining to the study area is to be overlain on the contour map, it too must be rotated and displaced in the same fashion as the array data. The purpose of ROTDXF is to effect this translation. (Note that no such rotation and translation is necessary if the finite-difference grid is oriented with its row direction east-west, for REAL2SRF then writes a SURFER grid file using real-world coordinates.)

Thus to use SURFER to contour model results directly for a finite-difference model in which the grid is uniform but not oriented with its row direction east-west, the following steps should be followed:

- Use REAL2SRF to generate a SURFER grid file based on a two-dimensional model results array.
- Use ROTDXF to rotate any DXF files containing map data that you wish to superimpose on the contour map.
- Import the grid and DXF files into SURFER and superimpose them.
- Use the SURFER "Arrange/Rotate" function to rotate the resulting map into its correct orientation. (SURFER-generated axes should be removed from the resulting diagram as the labels pertain to a rotated, translated coordinate system that has no real-world significance.)

Note that point-based information can also be rotated for posting on a map of this kind using program ROTDAT.


**See Also**

See also REAL2SRF, ROTBLN, ROTDAT.

# SECTION

**Function of SECTION**

SECTION interpolates the data contained in one or a number of MODFLOW/MT3D-compatible real arrays to points along an arbitrary line through the model domain. Interpolated results are written as data columns in a form suitable for input to commercial plotting software. Using such software a section of arbitrary complexity can be plotted through the finite-difference grid based on model results and aquifer/surface elevation data.

**Using SECTION**

A settings file `settings.fig` must be present in the directory from which SECTION is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

SECTION begins execution with the prompt:

```
 Enter name of grid specification file:
```

If a filename file (`files.fig`) was found in the current directory, a default grid specification filename may be included in the above prompt. If so, press <Enter> to accept the default or type in the appropriate filename.

Next SECTION prompts for the endpoints of a transect through the grid. A transect is defined by the eastings and northings (ie. *x* and *y* coordinates) of its two endpoints. These can lie in an arbitrary position; if the resulting transect line does not intersect an active part of the finite difference grid, SECTION will soon inform you. The transect line does not need to lie parallel to the row or column direction of the model grid; its orientation can be quite arbitrary. SECTION's prompts are:

```
 Enter end points of transect ----->
    east  coordinate of transect  start:
    north coordinate of transect  start:
    east  coordinate of transect finish:
    north coordinate of transect finish:
```

The interpolation density along the transect line is next selected at the prompt:

```
 Enter sampling interval along transect line:
```

It is suggested that a sampling interval of at most one fifth of the smallest cell dimension be entered here. Note that SECTION does not interpolate to points on the transect line that lie outside the finite difference grid or lie within inactive (or dry) cells; more on this below.

SECTION next requests the names of files holding the arrays which it must interpolate to points along the transect line. First it needs to know how many arrays require interpolation so that it can set aside memory to hold them. So it prompts:

```
How many arrays to be interpolated to section line?
```

to which you should respond with a non-negative integer. Then SECTION prompts:

```
Enter the names of the files containing real arrays ----->
   real array number 1:
   real array number 2:
          etc.
```

When supplying the name of a file holding a real array, keep in mind the naming convention outlined in Section 2.17 of Part A of this manual. If the supplied filename does not possess an extension of either "REF" or "REU", SECTION asks whether the file is formatted or unformatted, and reads the file accordingly.

SECTION does not interpolate to points lying within dry or inactive cells. It prompts:

```
 Enter blanking threshold value:
```

Any cell whose real array absolute value exceeds this blanking threshold is treated as dry or inactive; no interpolation takes place to points on the transect line situated within such cells. If an elevation array (for example the top of an aquifer) is included in the transect line, it may make a more pleasing pictorial effect if elevations outside the active grid area are assigned values above the blanking threshold; this can be achieved using program INT2REAL together with the activity (ie. *ibound*) array for the layer concerned.

SECTION provides two options for interpolating to points along the transect line which are within a half cell width of an inactive cell or the edge of the grid. The first option is not to interpolate to them at all, thus ignoring all points past the outer row/column of active cell centres. The second option is to interpolate to these points using an algorithm that accounts for missing cell centres; (see the documentation to program MODBORE from the PEST MODFLOW/MT3D Utilities). However as this algorithm does not extrapolate the trends from internal cells, the resulting section may appear artificially flat past the last row/column of active cell centres. Whether this happens or not will soon be apparent once the transect is plotted. The user can then choose the optimal scheme for the current model by responding appropriately to SECTION's next prompt:

```
 Interpolate to full grid or outer active cell centres? [f/c]
```

Respond with "f" if you want SECTION to interpolate right up to the edge of the active model area, or "c" otherwise.

SECTION next prompts for the name of the output file to which it should write its interpolated array data:

```
 Enter name for output file:
```

It then carries out its multiple array interpolation, writing the results to its output file. An example of a SECTION output file is shown below.

```
distance, easting,      northing,    surface.dat,  heads.dat
1060.000, 5.604295934, -664.798568, 31.45738,     26.45738
1070.000, 15.09112891, -661.636290, 31.54834,     26.45786
1080.000, 24.57796189, -658.474013, 31.67542,     26.45834
1090.000, 34.06479488, -655.311735, 31.84593,     26.45882
1100.000, 43.55162786, -652.149457, 32.02345,     26.45930
1110.000, 53.03846084, -648.987180, 32.25890,     26.45890
1120.000, 62.52529382, -645.824902, 32.33465,     26.45662
1130.000, 72.01212680, -642.662624, 32.45762,     26.45432
1140.000, 81.49895978, -639.500347, 32.65200,     26.45200
1150.000, 90.98579276, -636.338069, 32.84966,     26.44966
1160.000, 100.4726257, -633.175791, 33.03731,     26.44731
1170.000, 109.9594587, -630.013514, 33.14493,     26.44493
1180.000, 119.4462917, -626.851236, 33.34254,     26.44254
1190.000, 128.9331247, -623.688958, 33.54013,     26.44013
1200.000, 138.4199577, -620.526681, 33.63771,     26.43771
1210.000, 147.9067906, -617.364403, 33.73527,     26.43527
1220.000, 157.3936236, -614.202125, 33.83038,     26.43038
```

**Extract from a SECTION output file.**

The SECTION output file is arranged in comma-delimited columns. The first row of the file contains column headings, the first three headings denoting distance (measured from the start of the transect), and eastings and northings pertaining to interpolation points situated on the transect line. The following headers record the filenames (truncated if they are too long) of the arrays from which data within the column was interpolated. To plot a section through the model using a commercial plotting package, use the data in the first column to obtain the *x* coordinate of each plotting point, and the data in the fourth and higher columns to obtain plotting point *y* coordinates.

**Uses of SECTION**

The figure below illustrates a section through a groundwater model. Four curves are plotted, all with an abscissa representing distance along the line from the user-defined transect starting point. The data forming the lower two heavily-plotted lines were extracted from MODFLOW real arrays describing the top and bottom elevations of an aquifer layer; the top line was interpolated from a real array describing the land surface elevation. Data for the thin line was interpolated from a MODFLOW-generated solution array (extracted from an unformatted MODFLOW heads output file using program MANY2ONE); this line represents the computed head along the transect line through the aquifer.
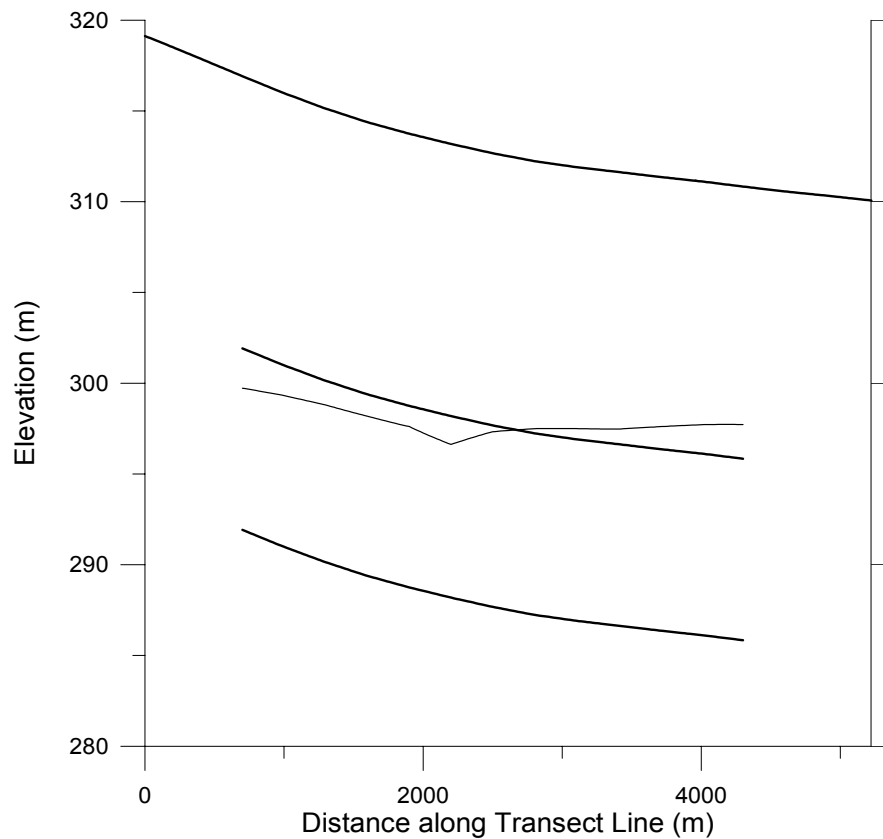
**Section through a single-layered model.**

As much complexity as a user desires can be added to a diagram such as the above. For example, multiple aquifers can be represented on the same figure, water levels at a number of different output times can be plotted, colour can be added, etc.

**See Also**

See also MANY2ONE.

# SMP2DAT

**Function of SMP2DAT**

Program SMP2DAT extracts information from a bore sample file for a user-specified list of bores, interpolating the sample data to a list of times corresponding to model output times. It generates a bore data file from which PEST input files can be constructed using programs from the PEST MODFLOW/MT3D Utilities suite.

Note that use of SMP2DAT has now been largely superseded the use of MOD2OBS and PESTPREP.

**Running SMP2DAT**

Program SMP2DAT will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities, and whether or not formatted integer and real arrays are preceded by a "number of columns, number of rows" header in the files in which they are stored.

SMP2DAT commences execution with the prompt:

```
 Enter name of bore sample file:
```

Respond with an appropriate filename. If the name of a bore sample file has been read from a filename file (`files.fig`) housed in the current directory, that filename will be presented on the above command line. In that case you should either accept the default by pressing <Enter> or type in the appropriate filename.

SMP2DAT needs to read a bore listing file in order to be informed of the bores for which it should time-interpolate bore sample readings. Enter the name of this file in response to the following prompt:

```
 Enter name of bore listing file:
```

Next SMP2DAT prompts:

```
 Enter name of MODBORE/MT3BORE-generated file:
```

SMP2DAT reads a MODBORE or MT3DBORE output file (see the PEST MODFLOW/MT3D Utilities) in order to determine the elapsed times for which model output is available. As is documented in the manual to the PEST MODFLOW/MT3D Utilities, comparisons can be made between model-generated numbers and their field counterparts only at those times at which MODFLOW or MT3D generates unformatted output data. The "model" actually run by PEST includes both MODFLOW (or MT3D) and MODBORE (or MT3BORE). PEST reads the output of either (or both) of the latter programs in order to determine model-generated heads, drawdowns or concentrations at boresites.

SMP2INFO must interpolate the data contained in the bore sample file to the model output times as provided in the MODBORE or MT3BORE output file. In order to do this it must know the starting date and time of the model simulation run and the time units used by the model. So it prompts:

```
Provide the following model timing data ---->
  Enter time units used by model (yr/day/hr/min/sec) [y/d/h/m/s]:
  Enter simulation starting date [dd/mm/yyyy]:
  Enter simulation starting time  [hh:mm:ss]:
```

Note that the starting date can be represented in the format mm/dd/yyyy if desired through appropriate editing of the settings file.

Next SMP2DAT requests that the user supply two parameters which govern the manner in which it interpolates sample data recorded in the bore sample file. The prompts are:

```
Provide the following time-interpolation parameters ---->
  Enter maximum days to nearest sample (fractional if necessary):
  Enter days over which sample can be assumed constant if linear interpolation
  cannot take place (fractional if necessary):
```

If a model output time is more than $y$ days removed from the nearest sample time for a particular bore, ($y$ being the user's response to the first of the above prompts) then no time-interpolation will take place to that model output time for that bore. Note that $y$ need not be an integer; it may be a fractional number (less than 1.0 if desired). The second parameter entered above informs SMP2DAT how to proceed if a model input time does not lie between two sample times for a particular bore, or if data at the further sample time is unacceptable (denoted by an "x" in the bore sample file). In this case, if a model output time is within $z$ days of a good sample, SMP2DAT will assign a value at that time equal to that of the closest sample. Note that like $y$, $z$ can be fractional (and less than unity).

Before it commences its processing of the bore sample file, SMP2DAT requests the name of the "bore data" output file which it must generate. For a full discussion of the format of this file and its role in generating files suitable for the use of PEST with MODFLOW and MT3D, see the manual for the PEST MODFLOW/MT3D Utilities suite. See also Section 2.3 of Part A of this manual. SMP2DAT prompts:

```
 Enter name for output bore data file:
```

Next it asks:

```
 Use numbers or bore identifiers for observation names?  [n/a]
```

If you choose the former alternative, observations are named 1,2,3,...99999999. The upper limit of 99999999 results from the fact that PEST restricts observation names to 8 characters in length. If you chose the bore identifier alternative, observation names are derived from the bore identifier name and the MODFLOW output time. The output time is expressed as an "_*nn*" suffix in the observation name where *nn* refers to the fact that the observation time corresponds to the *nn*'th time of array output in the MODBORE output file. The first part of the observation name is formed from either the first *m* or the last *m* letters of the bore identifier (user's choice through the ensuing prompt), where *m* is determined by SMP2DAT as long enough to allow both the bore name fragment and the output time

number to fit within the 8 characters comprising the observation name. If resulting observation names are not unique, SMP2DAT warns the user of this. Use of bore identifiers and output times to form observation names allows the user to mentally link observation names to measurements with greater ease than if numbers are used for observation names.

Note that if, for some reason, no time-interpolation can take place to any of the model output times for a bore listed in the bore listing file (for example if the bore is not cited in the bore sample file) SMP2DAT warns the user of this.

**Uses of SMP2DAT**

SMP2DAT's sole use is in preparation for a PEST run in which parameters are estimated for MODFLOW and/or MT3D. As such it is an extremely powerful tool, carrying out most of the work involved in the tedious process of extracting and formatting field data so that it is in a form where it can be used by PEST for comparison with corresponding model-generated numbers  When used with members of the PEST MODFLOW/MT3D Utilities suite, SMP2DAT reduces preparation time to a minimum.

Note that in many situations it is better to interpolate model-generated quantities to measurement times than to interpolate measured quantities to model output times. Program MOD2OBS carries out the former operation; PESTPREP can then be used for PEST input file generation. See Section 4.14 of Part A of this manual for further details.

**See Also**

See also MOD2OBS, PESTPREP, SMP2INFO and SMP2PM2.

# SMP2HYD

**Function of SMP2HYD**

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

**Using SMP2HYD**

Program SMP2HYD will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities.

SMP2HYD begins execution with the prompt:

```
 Enter name of bore sample file:
```

to which you should respond by typing an appropriate filename. If a filename file (`files.fig`) is present in the directory from which SMP2HYD was invoked a default bore sample file will appear with the above prompt. You can either accept the default by hitting <Enter>, or ignore it by providing a more appropriate filename.

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

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

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

When there are no further bores SMP2HYD asks:

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

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

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

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

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

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

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

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

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

```
TIME_IN_DAYS      DATE          TIME          BORE_13500006A
   -490.50      02/12/1965    12:00:00         18.5200
   -435.50      26/01/1966    12:00:00         18.5000
   -434.50      27/01/1966    12:00:00         18.5000
   -434.50      27/01/1966    12:01:01         18.0000
   -402.50      28/02/1966    12:00:00         18.4500
   -371.50      31/03/1966    12:00:00         18.4300x
   -322.50      19/05/1966    12:00:00         18.3000x
   -300.50      10/06/1966    12:00:00         18.3300x
   -275.50      05/07/1966    12:00:00         18.1500
   -182.50      06/10/1966    12:00:00         18.1200
   -119.50      08/12/1966    12:00:00         18.0000
   -048.50      17/02/1967    12:00:00         17.9200x
    -31.50      06/03/1967    12:00:00         17.9200x
      1.50      15/05/1967    12:00:00         17.9200x
     24.50      21/06/1967    12:00:00         18.5600
    102.50      13/07/1967    12:00:00         18.5300
    175.50      09/08/1967    12:00:00         18.6300
    217.50      06/10/1967    12:00:00         18.6300
    257.50      05/12/1967    12:00:00         18.5600
    323.50      08/01/1968    12:00:00         18.5600
    360.50      11/03/1968    12:00:00         19.2900
    393.50      17/05/1968    12:00:00         19.4700
```

**Extract from a SMP2HYD output file.**

The first column of a SMP2HYD output file lists time elapsed since the reference time; for samples prior to the reference time elapsed times are negative. (Note that the header to this column records the units used for elapsed time as previously supplied by the user.) The second and third columns list sample dates and times respectively; these were transferred directly from the bore sample file (reset file settings.fig to record dates in the mm/dd/yyyy format). The fourth column lists the sample value. Note that if a value was marked by an "x" in the bore sample file, the "x" is

transferred to the SMP2HYD output file. However instead of being in a column of its own, it is placed directly against the number which it denotes as suspect. This makes the sample value an invalid number. Some plotting packages, when they fail to read the number, will object with an error message; others will simply ignore the number, probably a desirable feature when the data is plotted. A user can search for the presence of suspect data in a SMP2HYD output file by simply importing the file into a text editor and searching for "x".

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

**Uses of SMP2HYD**

For cases where a bore sample file has been created by downloading data from a groundwater database, SMP2HYD provides the means whereby hydrographs can be generated from that data with the maximum flexibility. However a bore sample file can also be created by interpolating model-generated data to boresites (or cell centres); see program MOD2SMP. In this case SMP2HYD provides the means whereby time-plots of model results can be obtained. Where model outputs are interpolated to the same set of bores as that for which measured data exists, SMP2HYD provides a mechanism for the direct comparison of model-generated quantities with measured quantities for the same bores. This is best achieved by plotting model results and measurements on the same graph.

**See Also**

See also programs MOD2SMP, SMP2INFO.

# SMP2INFO

**Function of SMP2INFO**

SMP2INFO reads a bore sample file, time-interpolating the data contained in that file to a user-specified date and time. It writes its output to a bore information file, supplying bore eastings and northings as two of the columns in this file. A SMP2INFO output file is ready for immediate use by a contouring package such as SURFER.

**Using SMP2INFO**

Program SMP2INFO will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities.

SMP2INFO commences execution with the prompt:

```
 Enter name of bore coordinates file:
```

If a filename file (`files.fig`) is present in the subdirectory from which SMP2INFO is run, SMP2INFO provides a default bore coordinates file with the above prompt. The user can either accept this default by pressing the <Enter> key or type in an alternative filename. Note that the bore coordinates file read by SMP2INFO does not require a fourth column listing model layer numbers pertaining to each bore. If this column is present, SMP2INFO simply ignores it. For a detailed discussion of the bore coordinates file see Section 2.2 of Part A of this manual.

SMP2INFO next asks for a bore listing file.

```
 Enter name of bore listing file:
```

The bore listing file must provide a list of the bores for which information is required in the SMP2INFO output file. Each bore listed in the bore listing file must also appear in the bore coordinates file. If you wish, you may enter the name of the bore coordinates file here as that of a bore listing file; SMP2INFO simply ignores columns after the first. Do this if you would like SMP2INFO to provide information for all bores appearing in the bore coordinates file. See Section 2.5 of Part A of this manual for a description of the bore listing file.

Next SMP2INFO requests the name of the bore sample file which holds the information for which time-interpolation is desired.

```
 Enter name of bore sample file:
```

If the name of a bore sample file has been read from a filename file (`files.fig`) resident in the directory from which SMP2INFO was invoked, its name will appear as

a default name with the above prompt. You can accept the default by pressing the <Enter> key; alternatively type the name of another bore sample file.

Having been informed of the names of its three input files, SMP2INFO prompts the user for some information regarding how it is to interpolate the information contained in the bore sample file to the date and time which the user will specify shortly. Interpolation takes place in a linear fashion. However SMP2INFO will not carry out such interpolation if the user-supplied interpolation date and time (ie. the date and time to which interpolation is desired) is too far removed from an actual measurement date and time. Furthermore, it needs to know what to do if the interpolation date and time precedes or postdate all sample dates and times for a particular bore. Hence it prompts:

```
Enter the following time-interpolation parameters --->
  maximum days to nearest sample  [90.000]:
  days over which sample can be assumed constant if
  linear interpolation cannot take place  [3.0000]:
```

SMP2INFO will not calculate an interpolated measurement for a bore if the interpolation date and time (supplied below) is more than the "maximum days to nearest sample" from the nearest sample found on the bore sample file for that bore. SMP2INFO supplies a default value of 90 days for this interpolation parameter; press <Enter> to accept this, or supply an alternative. Note that fractional days can be supplied here. Thus if SMP2INFO was being used to extract data gathered in the early stages of a pump test where water levels are dropping fast, a suitable "maximum days to nearest sample" may be 1 minute (ie. $6.94 \times 10^{-4}$ days).

SMP2INFO cannot undertake linear interpolation to a user-supplied interpolation date and time for a particular bore if the interpolation date and time is not bracketed by samples pertaining to that bore in the bore sample file read by SMP2INFO. This will be the case if the interpolation time precedes or postdates the first or last sample for the particular bore, or if the sample to either side of the interpolation date and time has an "x" beside it in the bore sample file. However, if the interpolation time is close enough to the nearest sample time, SMP2INFO will simply provide the nearest sample value as the interpolated value. "Close enough" is defined as the user's response to the second of the above prompts. Once again, the quantity must be supplied in days, though fractional days are permissible. However it must not exceed the "days to nearest sample" and it must be greater than zero (enter a very small positive number to simulate zero if this is critical). Simply press <Enter> to accept the default value of 3 days.

Next SMP2INFO requests the interpolation date and time:

```
Enter interpolation date [dd/mm/yyyy]:
Enter interpolation time  [hh:mm:ss]:
```

Note that the date format used by SMP2INFO depends on the contents of the settings file `settings.fig`.

Then, after requesting a suitable name of its output file:

```
Enter name for output bore information file:
```

SMP2INFO asks:

```
 Record any uninterpolated bores to output file?  [y/n]:
```

If you answer "n" to this question SMP2INFO will not record information to its output file for any bore in the bore listing file for which linear time-interpolation is impossible. If you answer "y" SMP2INFO indicates those bores for which time-interpolation cannot take place by writing (in the place where the interpolated value should be) a simple explanation for its lack of ability to undertake the required interpolation. Thus if a bore listed in the bore listing file was not found in the bore sample file SMP2INFO records "not_in_sample_file" as its time-interpolated sample for that bore. If the interpolation time predates the first sample for a certain bore by more than the "days over which sample can be assumed constant if linear interpolation cannot take place", SMP2INFO records "before_first_sample", whereas if it postdates the last sample time by more than the same amount, it records "after_last sample" as the interpolated sample value. If the interpolation time exceeds the "maximum days to nearest sample" from the closest sample recorded in the bore sample file, the interpolated value for that bore is recorded as "too_long_to_nearest_sample", whereas if one of the samples bracketing the interpolation time for a certain bore is marked with an "x" in the bore sample file, its interpolated value is denoted as "x_affected".

The figure below shows an extract from a SMP2INFO-generated bore information file. (Specifications of a bore information file can be found in Section 2.4 of Part A of this manual.)

```
13600147A    432215.0000    7253084.0     1.56636
13600153A    429199.0000    7244641.0  not_in_sample_file
13600154A    423826.0000    7252456.0    25.8000
13600155A    428946.0000    7255713.0    10.0959
13600156A    428604.0000    7256573.0     9.32348
13600157A    428490.0000    7256880.0     9.04947
13600158A    435513.0000    7262205.0    0.192786
13600159A    435377.0000    7261282.0    0.452000
13600160A    435101.0000    7260296.0    0.170000
13600161A    434829.0000    7258357.0   -0.401399
13600162A    430338.0000    7257935.0    -1.42840
13600163A    431671.0000    7255389.0  not_in_sample_file
13600164A    429269.0000    7258422.0     5.86394
13600165A    438342.0000    7251391.0  not_in_sample_file
13600166A    438139.0000    7252774.0  not_in_sample_file
13600167A    436928.0000    7253814.0  not_in_sample_file
13600168A    439354.0000    7257393.0  not_in_sample_file
13600169A    433519.0000    7256321.0  before_first_sample
13600170A    437174.0000    7249017.0  not_in_sample_file
13600171A    433992.0000    7245588.0  not_in_sample_file
13600172A    431026.0000    7249480.0  not_in_sample_file
13600173A    440403.0000    7248539.0  not_in_sample_file
13600174A    434268.0000    7252264.0  before_first_sample
```

**Extract from a bore information file generated by SMP2INFO.**

**Uses of SMP2INFO**

In most cases the bore sample file read by SMP2INFO will have been downloaded from a groundwater database. The bore information file generated by SMP2INFO,

based on the contents of the bore sample file, is immediately readable by most contouring packages. Hence by running SMP2INFO at a number of historical interpolation times (these should be chosen to be as close as possible to actual sample times) it is easy to generate a series of files from which a corresponding set of contour maps can be constructed to demonstrate the behaviour of an aquifer system over time.

With the help of program MOD2SMP a bore sample file can be constructed based on model-generated groundwater data interpolated to the sites of user-specified bores. Using program SMP2INFO, files can be generated from which contour maps can be drawn based on model-generated heads interpolated to specific bores. Thus a comparison can be made between a contour map generated from actual borehole data, and model-generated data interpolated to the same bores; this is the best way to compare model and field data for calibration purposes. (See also program MODBORE from the PEST MODFLOW/MT3D Utilities suite.)

**See Also**

See also MOD2SMP, SMP2HYD, SMPCHEK.

# SMP2PM1

**Function of SMP2PM1**

SMP2PM1 reads a bore sample file. It extracts from this file data pertaining to a set of user-supplied bores between two user supplied times. It then reformats this data as a PMWIN (version 4) observation file. Once this data has been read into PMWIN, a visual comparison can be made between model-generated results and field data.

**Running SMP2PM1**

SMP2PM1 begins execution by requesting the name of a bore sample file:

```
Enter name of bore sample file:
```

to which you should respond with an appropriate filename. If SMP2PM1 was able to read the name of a bore sample file from a filename file (`files.fig`) housed in the directory from which it was run, this name will appear with the above prompt; in this case you should either press <Enter> to accept the default or type in the appropriate filename.

The layout of a PMWIN observation file depends on the nature of the data represented in the file. So SMP2PM1 next asks:

```
Does this file contain head, drawdown, concentration, compaction,
  preconsolidated head, or subsidence data?  [h/d/c/m/p/s]:
```

After you select the appropriate option, SMP2PM1 prompts for the name of a bore coordinates file and then a bore listing file. The latter file should contain a list of bores for which data is to be extracted from the bore sample file. (Note, however, that these bores are provided with different identifiers before being presented to PMWIN as discussed below.) The SMP2PM1 prompts are:

```
Enter name of bore coordinates file:
Enter name of bore listing file:
```

Next SMP2PM1 asks:

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

If you respond with "a", SMP2PM1 will record on its output file all samples pertaining to all bores cited in the bore listing file, except for those samples marked with an "x" (see Section 2.7 of Part A of this manual for the specifications of a bore sample file). However, if you respond with "w", SMP2PM1 will record only those samples that lie between two user-specified dates and times. These date and times are supplied to SMP2PM1 in response to the following prompts:

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

Note that the date format used by SMP2PM1 can be altered by editing the settings file `settings.fig` in the appropriate manner. SMP2PM1 then requests the reference time and date. This is the instant with respect to which times are measured; times prior to the reference time are negative while times following the reference time are positive. The prompt is:

```
 Enter reference date [dd/mm/yyyy]:
 Enter reference time   [hh:mm:ss]:
```

PMWIN plots the data contained in an observation file with elapsed time as the abscissa. Zero elapsed time corresponds to the beginning of a model run, ie. to the beginning of the first time step of the first stress period. Hence the reference time supplied above should be chosen as the time corresponding to the beginning of the model simulation period. The time units used by the model must also be supplied in order to ensure coincidence of field and model-generated data when plotted by PMWIN; hence SMP2PM1 next prompts:

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

The only other items of information now required by SMP2PM1 are the names of the files to which it should write its output data. These filenames are supplied in response to the following prompts:

```
 Provide the names for the following SMP2PM1 output files:-
 Enter name for output PMWIN observation file:
 Enter name for output PMWIN bore coordinates file:
 Enter name for output bore_id to PMWIN_id file:
```

It is strongly advised that extensions of "OBS", "BOR" and "ID" be provided for the above files (the first two are PMWIN defaults for the respective file types).

After accepting these filenames SMP2PM1 extracts data from the bore sample file, conveying it to the PMWIN observation file. If there are any bores cited in the bore listing file for which no information could be extracted from the bore sample file (eg. if the bore is not cited in the latter file, there is no data for the bore within the user-defined window, all of its samples are marked with an "x", etc), then SMP2PM1 generates an appropriate warning message and no information is written for that bore to the PMWIN observation file. Then, for all bores for which information is written to the PMWIN observation file, SMP2PM1 writes coordinates to a PMWIN bore coordinates file. This file can be read by PMWIN at the same time as the observation file; the information contained in it allows PMWIN to interpolate model results to these bores so that a comparison can be made between field and model-generated data. Note that if a bore cited in the bore listing file is not recorded on the PMWIN observation file due to the fact that no information could be supplied for it, the bore does not appear in the SMP2PM1-generated bore coordinates file.

Specifications for PMWIN observation and bore coordinates files are provided with the PMWIN documentation. Note that data extracted from the bore sample file is written to the 4th, 5th, 6th, 7th, 8th or 9th data column of the observation file depending on whether the user indicated that it contains head, drawdown, concentration, compaction, preconsolidated head or subsidence data.

PMWIN version 4 does not accept a 10 character bore identifier as do the Groundwater Data Utilities documented in this manual. It is the PMWIN version 4 convention that bores be numbered sequentially, starting at 1. Thus SMP2PM1 renames the bores cited in the bore listing file, starting at 1 and increasing in increments of 1. Note, however, that bores from the bore listing file which do not appear in the PMWIN observation file are not thus renamed.

In order that a user may easily identify bore data when plotted by PMWIN, SMP2PM1 provides a file listing correspondences between a user's borehole naming convention and that adopted by PMWIN; this is the third file referred to in the above series of prompts. An example of this type of file is shown below.

```
bore_identifier    PMWIN_bore_identifier
13500002A                 1
13500010A                 2
13500011A                 3
13500020A                 4
13500021A                 5
13500024A                 6
13500035A                 7
13500042A                 8
13500044A                 9
```

**A bore identifier file.**

As is obvious from the above illustration, the first column of the bore identifier file contains bore identifiers as supplied through the bore listing and bore coordinates files (converted to upper case). The second column contains the bore identifier supplied to PMWIN and is comprised of integers increasing from 1.

**Uses of SMP2PM1**

SMP2PM1 serves as a useful interface between a user's data and PMWIN. It allows a user to quickly and easily supply field data to PMWIN for comparison with model results using the PMWIN "Curves" facility. Version 4 of PMWIN can plot up to 6000 data points pertaining to up to 100 bores jointly with model results interpolated to the same bores. These plots are colour-coded according to bore, with field data plotted using thick lines and model data plotted as thin lines. Plots of this kind provide invaluable assistance when calibrating transient models either manually, or with the aid of PEST.

**See Also**

See also SMP2INFO, SMP2PM2.

# SMP2PM2

**Function of SMP2PM2**

SMP2PM2 carries out time-interpolation of the data contained in a bore sample file to those simulation times at which a model generates unformatted output. It records this data as a PMWIN (version 4) observation file. As such it provides an extremely powerful and rapid means of data preparation for a PEST run, complementing PMWIN's functionality as a PEST preprocessor.

**Running SMP2PM2**

Program SMP2PM2 will not run unless a settings file (`settings.fig`) is present within the directory from which it is invoked. As discussed in Section 2.19 of Part A of this manual, a settings file determines the manner in which dates are represented by the Groundwater Data Utilities.

Upon commencement of execution SMP2PM2 prompts for the name of a bore sample file:

```
 Enter name of bore sample file:
```

Note that if SMP2PM2 was able to read the name of a bore sample file from a filename file (`files.fig`) situated in the directory from which it was run, that filename would appear with the above prompt. Either press <Enter> to accept the default or enter the appropriate filename.

A PMWIN observation file can contain any of six different types of data. In order to generate a PMWIN observation file of the correct format, SMP2PM2 needs to know the kind of data represented in the bore sample file whose name was provided above. So it prompts:

```
 Does this file contain head, drawdown, concentration, compaction,
   preconsolidated head, or subsidence data?  [h/d/c/m/p/s]:
```

to which you should respond with the appropriate character. SMP2PM2 then requests the names of a bore coordinates and a bore listing file. The latter file should contain a list of bores for which information is required in the PMWIN observation file generated by SMP2PM2. The former file should contain borehole geographical coordinates. Note that while the Groundwater Data Utilities will accept any bore identifier of up to 10 characters in length in bore listing and bore coordinates files, these identifiers are not transferred to the PMWIN (version 4) observation file, other identifiers being used in their stead to satisfy PMWIN's bore identifier protocol; this is further discussed below. The prompts are:

```
 Enter name of bore coordinates file:
 Enter name of bore listing file:
```

A default bore coordinates filename will appear with the first of the above prompts if supplied within a filename file situated in the current directory. The default can be accepted by pressing the <Enter> key, or overridden by typing an alterative filename.

Next SMP2PM2 prompts:

```
 Enter name of MODBORE/MT3BORE-generated file:
```

SMP2PM2 needs to read a MODBORE or MT3BORE (from the PEST MODFLOW/MT3D Utilities) output file in order to inform itself of those times at which MODFLOW or MT3D generates unformatted output. When PEST is being used for MODFLOW parameter estimation in conjunction with PMWIN, the "model" which PEST runs is actually MODFLOW followed by MODBORE (or MT3D followed by MT3BORE). Under these circumstances the file written by MODBORE is named `MODELOUT.DAT`; this filename should be supplied in response to the above prompt. For the correct operation of the parameter estimation process in the PMWIN environment, field data must be supplied to PEST at times corresponding to the times at which the "model" generates data. See the manual for the PEST MODFLOW/MT3D Utilities for more details.

When PMWIN acts as a PEST preprocessor, it generates the instructions required to read a MODBORE output file. However it only generates the particular instruction required to read a specific model-generated number corresponding to a certain borehole observation when the time at which that observation was recorded corresponds to a time at which MODFLOW is instructed (through PMWIN) to produce unformatted output. This is why field data must be interpolated to model output times prior to importation into PMWIN.

In order that it may relate real time, as expressed in the bore sample file, to model simulation time, SMP2PM2 prompts:

```
Provide the following model timing data ---->
   Enter time units used by model (yr/day/hr/min/sec) [y/d/h/m/s]:
   Enter simulation starting date [dd/mm/yyyy]:
   Enter simulation starting time   [hh:mm:ss]:
```

Note that the date format used by SMP2PM2 depends on the contents of the settings file `settings.fig` housed in the current directory.

Before carrying out time-interpolation from the sample times recorded in the bore sample file to the model output times, two time interpolation parameters must be supplied. These are the same parameters required by program SMP2INFO which also carries out time-interpolation of a bore sample file. SMP2PM2 prompts:

```
 Provide the following time-interpolation parameters ---->
   Enter maximum days to nearest sample (fractional if necessary):
   Enter days over which sample can be assumed constant if linear interpolation
   cannot take place (fractional if necessary):
```

If a model output time is more than *y* days removed from a sample time for a particular bore (where y is supplied in response to the first of the above prompts), time-interpolation will not take place to that bore for that sample time; hence no data will appear for that bore at the corresponding model output time in the PMWIN observation file generated by SMP2PM2. Through proper choice of this time-interpolation parameter, a user is thus able to instruct SMP2PM2 to select only samples that are closer than a certain upper limit to model output times; the ensuing parameter estimation process will then be based only on such coincident (or nearly

coincident) model/field-data pairs. Note that fractional days can be supplied for this parameter.

The second of the above parameters governs SMP2PM2's behaviour if a particular model output time does not lie between two (non-x-affected) bore sampling times as recorded in the bore sample file. Normally time-interpolation cannot take place to such times. However if such a model output time is within a certain temporal distance of a sample time, SMP2PM2 assigns the time-interpolated value for that time as the value of the nearest sample. For example if a particular model output time falls one day short of the first sample time for a bore listed in the bore listing file and the above interpolation parameter is supplied as 2 days, the sample value recorded in the SMP2PM2-generated PMWIN observation file for that particular bore at that model output time will be equivalent to the bore's first sample.

The user must next supply names for SMP2PM2's output files:

```
Provide the name for the following SMP2PM2 output files:-
Enter name for output PMWIN observation file:
Enter name for output PMWIN bore coordinates file:
Enter name for output bore_id to PMWIN_id file:
```

It is wise to provide filename extensions of "OBS", "BOR" and "ID" respectively for these files; the first two are PMWIN defaults for files of these types.

The PMWIN observation file generated by SMP2PM2 records values for bores listed in the bore listing file (whose name was supplied above) interpolated to model output times. However if time-interpolation cannot take place for a particular bore and time, no corresponding entry is written to the SMP2PM2-generated PMWIN observation file.

The PMWIN bore coordinates file written by SMP2PM2 contains the coordinates of all bores for which samples were written to the SMP2PM2-generated PMWIN observation file. If, for a particular bore cited in the bore listing file, time-interpolation can take place at no model output times, that bore will not appear in the PMWIN bore coordinates file written by SMP2PM2.

The PMWIN observation and bore coordinates files generated by SMP2PM2 must be considered as a pair and should both be imported into PMWIN together (using the "Estimation/Bores and Observations" menu item). Bore identifiers provided to PMWIN differ from those provided in the bore listing file in order to fulfil version 4 of PMWIN's requirement that bores be identified by numbers arranged in increasing order. SMP2PM2 numbers bores in order of their occurrence in the bore listing file, omitting bores for which no time-interpolation can take place for any model output time.

In order to assist the user in recognising the plots pertaining to different bores when inspecting observations in the "Run/Curves" section of PMWIN, SMP2PM2 provides a third file which records the correspondences between user-supplied bore identifiers and those used by PMWIN. The example below shows such a file.

```
bore_identifier    PMWIN_bore_identifier
13500002A               1
13500010A               2
13500011A               3
13500020A               4
13500021A               5
13500024A               6
13500035A               7
13500042A               8
13500044A               9
```

**A bore identifier file.**

**Uses of SMP2PM2**

Use of SMP2PM2 can reduce setup time for a model calibration run using PEST from hours to seconds. The savings in time, effort and error-potential that can be made through the deployment of SMP2PM2 to extract and list those borehole measurements which can be used in the model calibration process are enormous. The recording and formatting of field data for comparison with model-generated numbers has always been the most laborious part of parameter estimation preprocessing.

Once time-interpolated field data has been imported into PMWIN it can be further manipulated as desired. For example individual bores can be turned on or off; the colour scheme for plotting can be altered, different weights can be assigned to different measurements, etc.

**See also**

See also MOD2OBS, PESTPREP, SMP2DAT, SMP2INFO, SMP2PM1.

# SMP2SMP

**Function of SMP2SMP**

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

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

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

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

**Using SMP2SMP**

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

SMP2SMP begins execution with the prompt:-

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

If a filename file (`files.fig`) is present within the directory from which SMP2SMP is run, and if that file contains the name of a default bore sample file, then

that filename will appear in the above prompt. The user should either press the <Enter> key to accept the default, of type in an alternative bore sample filename.

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

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

in response to which an appropriate filename should be supplied.

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

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

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

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

- A "bore sample file" need not necessarily contain bore data. The name is used here in accordance with the convention adopted by the Groundwater Data Utilities. In fact, data of any type may be contained within such a file, including river flow data, stream quality data, etc.

SMP2SMP next prompts:-

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

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

Finally SMP2SMP prompts for the name of the bore sample file which it must generate. After having been supplied with this name, it searches for bore identifiers represented in the observation bore sample file which are also represented in the model-generated bore sample file. If any of the samples pertaining to these identifiers fall within the model simulation time window, SMP2SMP interpolates model results

to the dates and times corresponding to the samples. It then writes a new bore sample file containing model-generated equivalents to field observations.

As was mentioned above, the observation bore sample file can contain measurements outside of the model simulation time span, and can reference bores (or measurement types) that are not cited in the model-generated bore sample file. In neither case is a corresponding model-generated sample represented in the SMP2SMP-produced bore sample file. Also, if a sample in the observation bore sample file is accompanied by an "x" in the final column indicating suspect data (see Section 2.7 of Part A of this manual), then SMP2SMP does not interpolate model results to this sample. In the unlikely event that a model-generated sample is "x-affected", that sample is not used in the temporal interpolation process; the preceding sample or the next sample is used instead.

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

**Uses of SMP2SMP**

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

SMP2SMP is used where a particular model executable program or postprocessor produces a bore sample file listing model results at model output times. Thus SMP2SMP can be run following programs MOD2SMP and BUD2SMP as part of a composite model.

**See Also**

BUD2SMP, MOD2OBS, MOD2SMP and PESTPREP.

# SMPCAL

**Function of SMPCAL**

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

**Using SMPCAL**

*Configuration Files*

As soon as SMPCAL commences execution it searches the current directory for a file named `settings.fig` in order to ascertain the protocol which it must use to represent dates. See Section 2.19 of Part A of the manual to the Groundwater Data Utilities for further details.

Optionally a file named `files.fig` may also be present in the current directory. If so, SMPCAL employs the name of the bore sample file cited in that file (if present) as the default name for the "standard" bore sample file - see below. See Section 2.8 of Part A of this manual for further details.

*Bore Sample Files*

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

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

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

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

- For each bore cited in the logger bore sample file, there should be at least two samples in the standard bore sample file within the time frame spanned by the first

and last readings for that bore in the logger bore sample file (or just slightly outside of that timespan - see below).

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

*What SMPCAL Does*

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

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

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

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

It is worth noting that the interpolation scheme used by SMPCAL to obtain notional logged values at standard measurement times is actually a linear extrapolation process using the two sample values either before or after the standard measurement time, depending on whether calibration coefficients are being sought for the preceding or following logged interval. The reason for this is that, for some logging systems, the downloading of logger data (which often accompanies manual measurement) results in an unfortunate "glitch", or offset, in logged values at the time at which these values are downloaded. By undertaking individual extrapolation from either side of the standard measurement point to obtain not one, but two, standard measurements, the effect of this extraneous offset can be "calibrated out".

*Running SMPCAL*

SMPCAL is run using the command:

```
smpcal
```

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

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

Note that, in common with other programs from the Groundwater Data Utilities, the user can "backtrack" at any time to the previous prompt by replying with an "e" (for "escape") to any particular prompt. Note also that, in accordance with the specifications of a bore sample file as set out in Part A of this manual, an "x" in the $5^{th}$ column a bore sample file signifies dubious data in the $4^{th}$ column. Such lines are ignored by SMPCAL, being used neither for data calibration (if occurring in the standard data file) nor for data adjustment (if occurring in the logger data file).

Next SMPCAL prompts:

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

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

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

Then its run record file:

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

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

```
Data adjustment for bore SA123:-


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

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

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

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

**Part of a SMPCAL report file.**

If, during its execution, SMPCAL encounters a problem with either of its input bore sample files it writes an appropriate error message to the screen and terminates execution. Normally only a single message is written, this being clearly visible upon termination of SMPCAL execution. However for certain types of error, SMPCAL continues execution until all of the logger sample file is processed, continuing to report further errors to the screen as necessary. To inspect all errors generated in this way, place the answers to the five SMPCAL prompts in a file (for example *smpcal.in*), one under the other, and run SMPCAL using the command:

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

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

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


**Uses of SMPCAL**

The most obvious use of SMPCAL is to calibrate logger data against manual readings taken over the time that the logger was operating. The fact that *m* and *c* (as determined by SMPCAL) can vary with time facilitates automatic logger data adjustment to real measurement datum, even where logger calibration drifts with time.

However this can also lead to serious errors in data adjustment if the user is not extremely careful, as the following example demonstrates.

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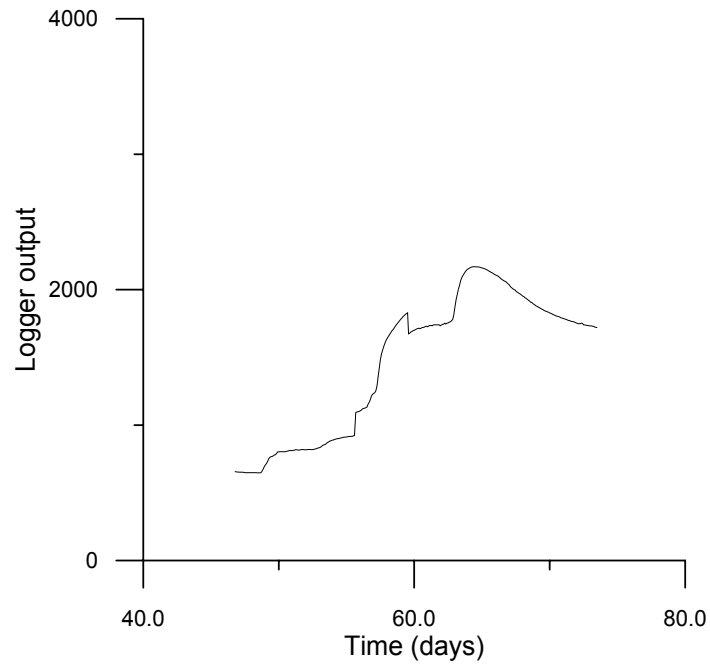


**Raw logger data plotted against time.**
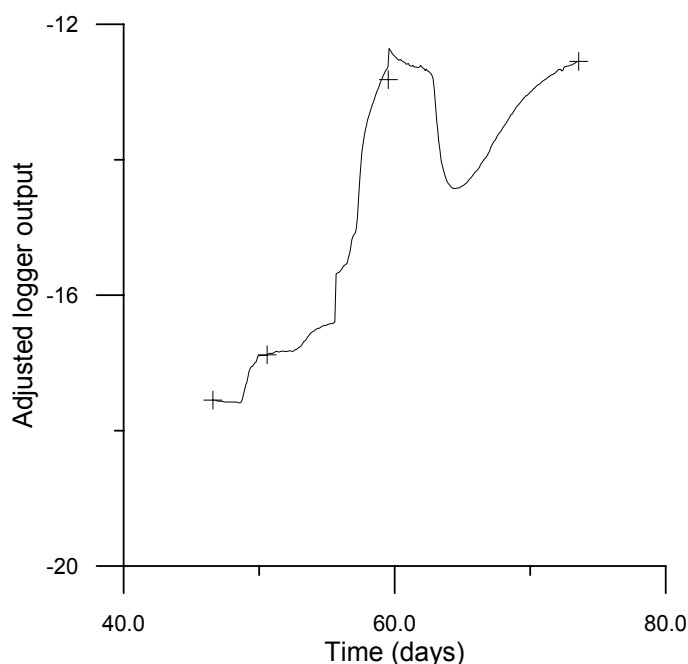
**Adjusted logger data plotted against time; adjustment carried out on basis of the four points shown.**

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

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

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

- always plot both the raw logger data and adjusted data,

- inspect the SMPCAL report file; significant variations of $m$ and $c$ between measurement data pairs is a good indicator or an unreliable calibration,

- it is best to use only a few standard data points spanning as high a vertical interval (ie. range of data values) as possible. **Neighbouring standard points should not have similar data values.**

- if possible, it is good practice to take two manual readings when visiting a site, one before downloading logger data and one after, in case the process of downloading the logger is responsible for any glitches.

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

When plotting adjusted data together with standard data as in the second of the above figures, standard data points will plot on the adjusted data curve. However the coincidence will not always be exact for, as has been mentioned above, where the time of a standard sample does not coincide with the time of a logged sample, the logged time-series is interpolated to the time of standard measurement. If there are rapid logger variations in the immediate vicinity of a manual reading (such as the glitch depicted above), the appearance that the standard reading does not plot exactly on the adjusted data results from the fact that the "join" between two neighbouring data intervals (in which different $m$ and $c$ values were used for adjustment) lies between two logged data points. The plotting package used to plot the adjusted data uses straight lines to join neighbouring logged data points; it does not deviate to account for the notional data value at the joining point. The latter, if it were determined by extrapolation from each neighbouring data time interval, would indeed plot at the same location as the standard point.

**See Also**

See also SMP2HYD.

# SMPCHEK

**Function of SMPCHEK**

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

**Running SMPCHEK**

Upon commencement of execution, SMPCHEK prompts:

```
 Enter name of bore sample file:
```

If the name of a bore sample file has been read from a filename file (`files.fig`) resident in the directory from which SMP2INFO was invoked, its name will appear as the default name with the above prompt. Accept the default by pressing the <Enter> key or type in the name of an alternative bore sample file.

SMPCHEK then reads the bore sample file, checking:

- that every line of the file has sufficient entries.

- that all numbers, dates and times are readable.

- that all dates and times are legal.

- that bore identifiers are 10 characters or less in length.

- that all entries for each bore are consecutive and are in order of increasing date and time.

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

**Uses of SMPCHEK**

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

Once a bore sample file has been created either by directly downloading a file from a groundwater database or by performing some elementary processing of a database-downloaded file (using programs such as MKSMP1), the file should be checked with SMPCHEK to establish its integrity. Errors can then be corrected all at once so that use of the file in subsequent processing will hold no surprises.

**See Also**

See also MKSMP1, SMP2HYD, SMP2INFO.

# SRF2REAL

**Function of SRF2REAL**

SRF2REAL rewrites a SURFER grid file as a MODFLOW/MT3D-compatible real array. This array can then be used by a model whose finite difference grid is uniform and whose cell centres coincide with the nodes of the SURFER grid. In this way SURFER's powerful gridding algorithms can be used to create model arrays from randomly-distributed data points.

**Using SRF2REAL**

A settings file `settings.fig` must be present in the directory from which SRF2REAL is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

SRF2REAL begins execution with the prompt:

```
 Enter name of grid specification file:
```

to which you should respond by typing in an appropriate filename. If SRF2REAL was able to read the name of a grid specification file from a filename file (`files.fig`) located within the current directory, the name of that file will appear with the above prompt. In this case you should press the <Enter> key to accept the default or type in the correct filename.

Before it prompts for any further information SRF2REAL reads the grid specification file to verify that the finite difference grid is uniform. If it is not uniform it cannot be matched to a SURFER grid; in such a case SRF2REAL terminates execution with an appropriate error message.

SRF2REAL then asks for the name of a SURFER grid file:

```
 Enter name of SURFER grid file:
```

Here you must enter the name of an ASCII SURFER grid file. Note that SURFER's default condition is to write a binary grid file; a user must explicitly tell SURFER to generate an ASCII file. To do this, click on the "Change" button in the "Output Grid File" section of the "Scattered Data Interpolation" dialogue box.

After it has read the SURFER grid file and verified that its dimensions are the same as those of the finite-difference grid, SRF2REAL prompts:

```
 Translate blanked grid values to what number?
```

SURFER uses the value $1.70141 \times 10^{38}$ to denote a blanked node. It is a good idea to translate this to a number of similarly high magnitude (for example $10^{30}$) when building the MODFLOW/MT3D real array. Such high values are then easily recognised as "dummy" values, and can be treated as such by supplying an

appropriate "inactive threshold" to many of the Groundwater Data Utilities documented in this manual.

Finally SRF2REAL prompts:

```
 Enter name for real array file:
```

Enter an appropriate filename, keeping in mind the naming convention for real arrays discussed in Section 2.16 of Part A of this manual. If the filename you provide does not possess an extension of "REF" or "REU", SRF2REAL prompts:

```
 Write a formatted or unformatted file?  [f/u]
```

Enter "f" or "u" as desired. An unformatted file cannot be read or edited using a text editor; however it requires less storage space and can still be read by the Groundwater Data Utilities documented in this manual.

**Uses of SRF2REAL**

When a finite-difference grid is uniform, a model array can be represented in SURFER grid file format; similarly, the contents of a SURFER grid file can be written in model real array file format. Hence SURFER can be used to prepare a two-dimensional model data array by spatial interpolation from random data points.

It is the user's responsibility to ensure that the intersection points (ie. nodes) of the SURFER grid overly the cell centres of the model grid. Thus a SURFER grid must have the same number of rows and columns as the model grid, and *x* and *y* direction spacings must equal model grid row and column direction spacings. The top left node of the SURFER grid must coincide with the top left cell centre of the model grid; note that the coordinates of the latter point must be distinguished from those of the top left corner of the model grid itself as recorded in the grid specification file. The former coordinates must be calculated from the latter; add half a cell width (ie. *delr/2.0*)in the *x* direction and subtract half a column width (ie. *delc/2.0*)in the *y* direction (*delr* is the model grid row direction width while *delc* is its column direction width).

If a model grid is not oriented with its row direction east-west and its column direction north-south, superimposition of a SURFER grid on a model grid is more difficult because SURFER grids must be oriented with their *x* direction pointing east and their *y* direction pointing north. However superimposition is still possible if the random data points used for interpolation to the SURFER grid are rotated about the top left corner of the finite difference grid and the top left corner of the grid is assigned coordinates of (0,0). After such rotation and translation (performed automatically by program ROTDAT), the *x* and *y* coordinates of the random data points are then expressed in terms of a coordinate system in which (0,0) corresponds to the top left hand corner of the finite difference grid, the *x* direction increases along the model grid row direction in the direction of increasing column number and the *y* direction coincides with the model grid column direction, *y* increasing with decreasing row number. If a SURFER grid is then specified whose *x* direction width is equal to the model row width, whose *y* direction width is equal to the model column

width, and whose top left node is situated at (*delr/2.0,-delc/2.0)*, such a SURFER grid will match the uniform model grid.

**See Also**

See also REAL2SRF, ROTBLN, ROTDAT, ROTDXF.

# TABCONV

**Function of TABCONV**

TABCONV converts between the two different types of cell representation used in real and integer array table files. These are cell identification by row and column number and cell representation by cell number. TABCONV rewrites an integer or real array table file which uses one of these cell identification conventions as an integer or real array table file which uses the alternative convention.

**Using TABCONV**

TABCONV begins execution with the prompt:

```
 Enter name of grid specification file:
```

Enter the appropriate filename. Alternatively, if a filename file (`files.fig`) is present within the current directory, a default grid specification filename may appear with the above prompt; press <Enter> to accept the default or enter the appropriate name. TABCONV needs to read the grid dimensions from the grid specification file so that it can convert between cell representations using row and column format and those using cell number format.

Next TABCONV asks:

```
 For cell identifier translation:
   if from row/column number to cell number  -  enter 1
   if from cell number to row/column number  -  enter 2
 Enter your choice:
```

Enter "1" or "2" as appropriate. If the first option is selected TABCONV reads an integer or real array table file (see Sections 2.11 and 2.18 of this manual) comprised of three data columns, the first two of which contain cell row and column numbers; the third column lists integer or real array values pertaining to the cells identified in the first two columns. It writes an integer or real array table file comprised of two data columns, the first containing cell numbers and the second containing integer or real array values. Cell numbers provide a unique cell representation; they are obtained by counting cells, row by row, from the top left corner of the finite difference grid. On the other hand if the second of the above options is selected, TABCONV reads an integer or real array table file comprised of two data columns, the first column containing cell numbers and the second containing integer or real array values; it writes an output integer or real array table file comprised of three data columns. In both cases the integer or real array information contained in the last data column remains unchanged.

Next TABCONV requests the names of its input and output files:

```
 Enter name of array table input file:
 Enter name for array table output file:
```

Supply appropriate filenames. TABCONV then reads the nominated input files, carrying out the appropriate cell identifier translation, and writes the translated information to the nominated output file.

**Uses of TABCONV**

As explained in Section 4.6 of Part A of this manual, the Groundwater Data Utilities provide a rudimentary GIS interface. Programs INT2MIF and REAL2MIF provide the ability to write integer and real array data in MAPINFO Interchange Format; the MID file of the pair contains an integer or real array table in which cells are represented in row and column number format. Once imported and edited in MAPINFO, such an array can be downloaded as a similar three column table to be subsequently reformatted as an integer or real array using program TAB2INT or TAB2REAL.

An even more rudimentary interface to ARCINFO is provided by program GRID2ARC. GRID2ARC writes an ARCINFO "generate" file based on an "active" window of the finite-difference grid. Within this generate file, cells are identified by cell number. If any array data is to be imported into ARCINFO to accompany the "generate" file which supplies grid geographical information, this data must be supplied separately as an integer or real array table file. In this file, cells must be identified by cell number so that they can be linked to the geographical cell information provided in the "generate" file. The easiest way to achieve this is to build a MID file based on the same "active" part of the finite difference grid, and then to re-write this file in cell number format using program TABCONV.

Similarly, integer or real array data exported from ARCINFO in cell number format can be converted to row and column number format by TABCONV before being re-written in true integer or real array format by program TAB2INT or TAB2REAL.

**See Also**

See also GRID2ARC, INT2MIF, REAL2MIF, TAB2INT, TAB2REAL.

# TAB2INT

**Function of TAB2INT**

TAB2INT rewrites the contents of a three-column integer array table file as a MODFLOW/MT3D-compatible integer array file. It provides the means whereby integer array data, previously uploaded to a GIS for graphical editing, can be reassimilated into a groundwater model.

**Using TAB2INT**

A settings file `settings.fig` must be present in the directory from which TAB2INT is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

Upon commencement execution TAB2INT prompts:

```
 Enter name of grid specification file:
```

Respond to this prompt with an appropriate filename. Alternatively, if TAB2INT has read the name of a grid specification file from a filename file (`files.fig`) housed in the current directory, that grid specification filename will appear with the above prompt. Press <Enter> to accept the default or type in the correct filename.

Next TAB2INT prompts:

```
 Enter name of integer array table file:
```

After you have provided a suitable filename TAB2INT prompts:

```
 Enter value to assign to uncited cells:
```

As explained in Section 2.11 of Part A of this manual, it is not necessary that every cell within a model grid be cited within an integer array table file. Where a cell has not been cited, TAB2INT assigns its value as the integer that you supply in response to the above prompt.

TAB2INT next reads the integer array table file, reporting any errors that it finds to the screen. If the integer array table file is error-free TAB2INT then prompts:

```
 Enter name for output integer array file:
```

Enter the name of the file to which the TAB2INT-generated integer array is to be written; keep in mind the integer array file naming conventions outlined in Section 2.10 of Part A of this manual.

**Uses of TAB2INT**

A model integer array can be stored within a geographical information system as a collection of "regions" or "polygons", each such region representing an individual model cell. Each cell can be assigned the "attributes" of row number, column number and integer array value. An integer array can be imported into a GIS using program INT2MIF documented elsewhere in this manual. After editing of integer array values (row and column numbers should not be edited) the integer array can be downloaded from the GIS as an integer array table. Program TAB2INT allows the data within this table to then be rewritten as a model-compatible integer array for subsequent model usage or model preprocessing.

**See Also**

See also INT2MIF, TAB2REAL.

# TAB2REAL

**Function of TAB2REAL**

TAB2REAL rewrites the contents of a three-column real array table file as a MODFLOW/MT3D-compatible real array. It provides the means whereby real array data, previously uploaded to a GIS for graphical editing, can be reassimilated into a groundwater model.

**Using TAB2REAL**

A settings file `settings.fig` must be present in the directory from which TAB2REAL is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

As it commences execution TAB2REAL prompts:

```
 Enter name of grid specification file:
```

Respond to this prompt with an appropriate filename. Alternatively, if TAB2REAL has read the name of a grid specification file from a filename file (`files.fig`) housed in the current directory, that grid specification filename will appear with the above prompt. Press <Enter> to accept the default or type in the correct filename.

Next TAB2REAL prompts:

```
 Enter name of real array table file:
```

After you have provided a suitable filename TAB2REAL prompts:

```
 Enter value to assign to uncited cells:
```

As explained in Section 2.18 of Part A of this manual, it is not necessary that every cell within a model grid be cited within a real array table file. Where a cell has not been cited, TAB2REAL assigns its value as the number that you supply in response to the above prompt.

TAB2REAL next reads the real array table file, reporting any errors that it finds to the screen. If the real array table file is error-free TAB2REAL then prompts:

```
 Enter name for output real array file:
```

Enter the name of the file to which the TAB2REAL-generated real array is to be written; keep in mind the real array file naming conventions outlined in Section 2.17 of Part A of this manual.

**Uses of TAB2REAL**

A model real array can be stored within a geographical information system as a collection of "regions" or "polygons", each such region representing an individual model cell. Each cell can be assigned the "attributes" of row number, column number and real array value. A real array can be imported into a GIS using program REAL2MIF documented elsewhere in this manual. After editing of real array values (row and column numbers should not be edited) the real array can be downloaded from the GIS as a real array table. Program TAB2REAL allows the data within this table to then be rewritten as a model-compatible real array for subsequent model usage or model preprocessing.

**See Also**

See also REAL2MIF, TAB2INT.

# TWOARRAY

**Function of TWOARRAY**

Program TWOARRAY constructs a MODFLOW/MT3D-compatible real array from two other such arrays. The other two arrays can be combined through addition, subtraction, multiplication, division or partial replacement. All of these operations can be restricted to a subset of each input array through the definition of active array subgroups on the basis of cell threshold values.

**Using TWOARRAY**

A settings file `settings.fig` must be present in the directory from which TWOARRAY is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

TWOARRAY commences execution with the prompt:

```
 Enter name of grid specification file:
```

Supply the name of the grid specification file appropriate to your current model. Note that if TWOARRAY found a filename file (`files.fig`) in the current directory it will display a default grid specification filename as part of the above prompt. If this happens, press <Enter> to accept the default filename or type in the appropriate grid specification filename.

Next TWOARRAY must read two real arrays. First it prompts:

```
 Enter name of first real array file:
```

Enter an appropriate filename, bearing in mind the convention for real array filename nomenclature discussed in Section 2.17 of Part A of this manual. Once the first real array has been read TWOARRAY prompts:

```
 Enter inactive threshold for first array (press <Enter> if none):
```

Here supply a single number or press the <Enter> key. The role of the inactive threshold value will be discussed shortly. TWOARRAY next prompts for the second real array filename and its inactive threshold value, if any:

```
 Enter name of second real array file:
 Enter inactive threshold for second array (press <Enter> if none):
```

Note that the second real array filename can be the same as that of the first, allowing the same array to be read twice. Next TWOARRAY asks how the two arrays are to be combined:

```
 Specify effect of second array on first array:-
 (replace/add/subtract/multiply/divide)  [r/a/s/m/d]:
```

TWOARRAY produces a third real array from the first two by first copying the first to the third and then performing one of the five operations listed above. The operation

is performed on a cell-by-cell basis. However if, for any cell, the absolute value of the first array exceeds the user-supplied first array threshold, or the absolute value of the second array exceeds the user-supplied second array threshold, the value of the pertinent third array cell remains as the value of the first array for the corresponding cell. (Note that if no array threshold has been supplied for an array, its threshold is set at infinity.)

If a user enters "r" at the above prompt, the second array replaces the first within the active window defined by both array threshold values as discussed above. If "a" is entered, the second array is added to the first, while if "s" is selected the second array is subtracted from the first (within the active window). If "m" is chosen array elements are individually multiplied while if "d" is typed, the first array is divided by the second; in the latter case make sure that the second array has no zero-valued elements within the active window as defined by both array threshold values.

After TWOARRAY has carried out the necessary array manipulation it prompts for the name of a file to which the new array should be written:

```
 Enter name for output real array file:
```

Enter a suitable filename, taking note of the real array filename conventions outlined in Section 2.17 of Part A this manual.

**Uses of TWOARRAY**

Most model preprocessors provide functionality to carry out mathematical operations between arrays. However the ability to restrict the application of these operations to user-specified zones within an array is not as common. This is unfortunate as it is a particularly useful aspect of MODFLOW preprocessing.

For example, a model may have been calibrated on the basis of a recharge array reflecting existing land use. If the model is then used to predict the effects of altering land use (and hence recharge) over part of the study area, one or a number of "partial recharge arrays" could be constructed in which a new recharge is assigned to areas where the proposed land use change is to take place; high cell values (ie. above the inactive threshold) are assigned to recharge cells elsewhere. These partial recharge arrays can then be "pasted" onto the original recharge array, leaving the latter unchanged in those areas where no land use changes are to take place.

**See Also**

See also INT2REAL, REAL2INT.

# VERTREG

**Function of VERTREG**

VERTREG adds "vertical regularisation" to a PEST control file in which at least some parameters are based on pilot points. For a multi-layer model VERTREG may need to be run a number of times; each time that it runs it provides linkages between pilot points assigned to one model layer and those assigned to another model layer. These linkages are formed through prior information equations. Where parameters are log-transformed, these prior information equations state that the preferred *ratio* of parameter values attached to pilot points in respective layers is equal to a user-supplied value. That is, the logs of parameter values are assigned a user-supplied preferred *difference*. Where parameters are untransformed, the prior information equations introduced to the PEST control file by VERTREG assign the preferred *difference* directly to pairs of parameter values from the different layers. This ratio or difference can be zone-dependent if desired.

It is assumed that the prior information equations added by VERTREG to the parameter estimation process will be used for regularisation purposes (probably adding to regularisation equations that are already present within the existing PEST control file). They can all be assigned the same weight (which is modified by the PEST-calculated regularisation weight factor through the course of the parameter estimation process), or they can be assigned different weights, with weights increasing with distance between pertinent pilot points and locations within the model domain at which measurements were taken for use in the calibration process. Thus regularisation can be enforced more strongly where calibration data density is low; this can greatly enhance numerical stability of the regularised inversion process.

**Using VERTREG**

*Further Background*

Where pilot points are used as a basis for spatial parameter definition, "horizontal" or "within-layer" regularisation can take a number of forms. One common form is "smoothness regularisation" where prior information is used to assign preferred values to parameter differences; see the PPKREG utility documented in this manual for details. Another option is "preferred value regularisation" where each parameter is individually assigned a preferred value; recognition of spatial correlation between parameter values can then take place by assignment of a covariance matrix (possibly based on the variogram for the area) instead of weights to the prior information equations which supply these preferred parameter values. When used in regularisation mode, PEST enforces these smoothness or preferred value relationships as strongly as it can without violating the need for model outputs to fit field data to within a user-specified tolerance (PEST variable PHIMLIM). Thus any heterogeneity introduced to a model domain, or any departure from preferred parameter values, exists because it MUST exist to calibrate the model.

As is explained in the PEST manual, regularisation observations and/or regularisation prior information equations are assigned to one or a number of observation groups whose names begin with the string "regul". The use of different observation groups for regularisation information allows PEST, or the user, to enforce regularisation conditions more strongly for some parameters than for others. Ideally, regularisation conditions should be applied more strongly to those parameters for which the information content of the calibration dataset is low than to those parameters for which it is high. Thus regularisation observations or prior information equations pertaining to "data-insensitive" parameters should be placed into regularisation groups which are provided with a higher weight than other regularisation groups. Alternatively, PEST can assign a higher weight to these groups automatically using its adaptive regularisation functionality.

When calibrating a multi-layer model, "vertical regularisation" may be needed in addition to "horizontal regularisation". This will be particularly important if horizontal regularisation is comprised solely of "smoothness regularisation". Smoothness regularisation ensures that parameter values within a particular aquifer or aquitard (for example $K_h$ and/or $K_z$ parameters within a particular model layer) vary horizontally only to the minimum extent required to obtain a good model-to-measurement fit. However use of smoothness regularisation does not prevent the parameter values for an entire layer from varying up or down as a whole. In many cases of multi-layer model calibration, this remaining variability after regularisation constraints have been imposed creates numerical problems for the inversion process because it prevents the achievement of a unique solution to the multilayer parameter estimation problem. This is unfortunate because the achievement of a unique solution is the very reason why regularisation is introduced to the parameter estimation process in the first place. Fortunately, this situation can often be rectified by the introduction of parameter linkages between different model layers. VERTREG allows these linkages to be introduced in the form of prior information equations in which parameter differences between layers (or the logs of parameter differences) are assigned preferred values. Where a value of zero is assigned, interlayer parameter equality becomes the "preferred regularisation condition"; this is then enforced to the maximum extent possible during the regularised inversion process while not compromising model-to-measurement fit.

Where parameterisation is based on pilot points, and where pilot points are placed at the same horizontal locations within different model layers, it is a simple matter to provide a set of prior information equations linking parameter values assigned to pilot points in one particular layer to parameter values assigned to corresponding pilot points in another layer. However in many modelling contexts, pilot points are placed at different horizontal locations within different model layers (and different numbers of such points may reside in different layers). In this case VERTREG provides linkages between pairs of points which are closest to each other. Where there are different numbers of pilot points in different layers (perhaps with greater pilot point density in one layer than in another), a particular pilot point in one layer may require linkage to more than one pilot point in another layer. VERTREG accommodates this situation, introducing the minimum number of vertical linkages required to ensure that every point in each layer is linked to at least one point in the other layer. However VERTREG can be instructed not to make a vertical linkage if the horizontal distance

between a particular pilot point in one layer and its closest neighbour in another layer is greater than a user-supplied distance threshold; thus no vertical regularisation is associated with that point. This can be useful in situations where one model layer has a larger areal extent than its underlying or overlying layer; points in that part of the broader layer which do not overly or underlie the neighbouring layer are then not linked to points in the latter layer.

Normally vertical linkages should be constructed between $K_h$ parameters in neighbouring layers, and between $K_v$ or vertical conductance parameters in neighbouring layers. As stated above, this vertical regularisation will normally be employed in addition to horizontal, within-layer regularisation for these same parameters.

*Running VERTREG*

As is the usual protocol for members of the Groundwater Data Utilities, the user may respond to any of VERTREG's prompts by typing "e" (for "Escape") followed by <Enter>. VERTREG will then return to its previous prompt, thus allowing the user to "backtrack" if he/she has made a mistake.

VERTREG commences execution with the prompt:-

`Enter name of existing PEST control file:`

It is assumed that this file is an error-free control file (check it with PESTCHEK to be sure) in which PEST is instructed to run in regularisation mode. It is thus assumed that the PESTMODE variable is set to "regularisation" and that the PEST control file contains a "regularisation" section. However it is not necessary for any prior information to exist within this file; on many occasions, however, prior information will, indeed, be present within the file as a result of the previous use of PPKREG or VERTREG (possibly on many occasions).

Next VERTREG prompts:-

`Enter name of pilot points file for one model layer:`

It is assumed that the PEST control file which VERTREG will modify cites parameter values pertaining to more than one model layer. (It can also cite different parameter types for each of these model layers.) In response to the above prompt the user must provide the name of a pilot points file which contains the names, locations, zones and values for pilot points used in the parameterisation of one of the model layers. The format of a pilot points file is described in Part A of this manual.

A single pilot points file can serve as the parameterisation basis for more than one model layer. As is described in the documentation of program PPKREG, different parameters based on the same set of pilot points are distinguished from each other in a PEST control file by use of a different parameter prefix; this prefix is affixed to the front of the name of each pilot point in forming the name of the parameter pertaining to that point for a particular layer (and/or for a particular parameter type). VERTREG needs to know this prefix; so it prompts:-

`Enter prefix for corresponding parameter names (<Enter> if none):`

Then VERTREG asks the same questions for another model layer, this layer being the one to which the parameters pertaining to the first model layer are to be linked. (Note that this second model layer can use the same set of pilot points, cited in the same pilot points file, if desired.) VERTREG prompts:-

```
Enter name of pilot points file for another model layer:
Enter prefix for corresponding parameter names (<Enter> if none):
```

Note that it is not necessary that every point represented in a pilot points file have a corresponding parameter in the PEST control file. Note also that it does not matter whether some pilot point parameters are tied or fixed in the existing PEST control file; VERTREG will detect this condition and refrain from supplying prior information for such tied or fixed parameters in accordance with the specifications of a PEST control file.

Next VERTREG either prompts:-

```
Will regularised vertical property ratio depend on zones? [y/n]:
```

or

```
Will regularised vertical property difference depend on zones? [y/n]:
```

The choice of "difference" or "ratio" depends on whether pilot point parameters are log transformed or untransformed in the PEST control file. For logarithmically transformed parameters a ratio is required; for untransformed parameters a difference is required (VERTREG ensures that ALL pilot point parameters associated with the two pilot points files under consideration are log transformed or untransformed; if some are log transformed and others are untransformed VERTREG will cease execution with an appropriate error message.) For the remainder of the explanation of VERTREG's behaviour presented below, it will be assumed that parameters are log transformed, and that a ratio is required rather than a difference; it should be noted, however, that some of VERTREG's prompts will be slightly different from those shown below where pilot point parameters are untransformed.

The user has the choice of supplying a single parameter ratio (applicable to all pilot points pertaining to the two model layers under consideration), or parameter ratios that vary with pilot point location. For the former option the user should respond to the above prompt with "n"; VERTREG then asks:-

```
Enter (first/second) parameter ratio for vertical regularisation:
```

This is the ratio obtained by dividing parameter values in the first model layer by those in the second model layer. Where parameter equality is the preferred regularisation condition, enter this ratio as "1"; in this case VERTREG will supply a suite of prior information equations specifying that the differences between the logs of linked pilot point parameters are all zero.

Alternatively, the user can specify that the vertical property ratio is zone-specific (by answering "y" to the above prompt). In this case VERTREG prompts:-

```
From which pilot points file is zonation taken?
     enter 1 if from file points1.dat:
     enter 2 if from file points2.dat:
Enter your choice:
```

As is discussed in Part A of this manual, a pilot points file specifies the zone to which each pilot point belongs. Pilot points in different layers can be assigned to different zones. It is the user's choice whether the zonation used to specify the parameter ratio (or difference) is taken from the first or second pilot points file; a point from each such file will be represented in each prior information equation introduced to the parameter estimation process by VERTREG. The user's selection is specified through the response supplied to the above prompt. After reading this response, VERTREG sorts the zone numbers cited in the selected pilot points file into ascending order and, for each zone, prompts:-

```
Enter (first/second) regularisation parameter ratio for zone n:
```

As discussed above, VERTREG links each pilot point in one layer to the closest point in the other layer. However if the horizontal distance between a particular pilot point in one layer and its nearest neighbour in the other layer is greater than a user-specified "exclusion distance", that point remains unlinked. The exclusion distance must be supplied in response to the prompt:-

```
Enter lateral exclusion distance:
```

Next VERTREG prompts:-

```
Enter name for new regularisation subgroup:
```

As was mentioned above, the PEST control file that is supplied to VERTREG must inform PEST that it is to be run in regularisation mode. Hence, as is explained in the PEST manual, there must be one or a number of observation groups present within this file whose names begin with "regul". Another such group must be introduced to the PEST control file to house the new set of vertical regularisation prior information equations constructed by VERTREG. So in response to the above prompt you should supply a name for this new observation group, ensuring that its name begins with "regul", and that it is no greater than 12 characters in length (as is the PEST convention). You should also ensure that the name does not conflict with the name of an existing observation group; VERTREG will inform you if such a conflict occurs.

As is discussed in the PEST manual, every prior information equation must have a unique name (of 12 characters or less in length). VERTREG provides a name for each new prior information equation by appending a number (starting from 1) to a base name which you should supply to VERTREG in response to the prompt:-

```
Enter basename for new prior information:
```

This name must be 6 characters or less in length (to leave room for high numbers). Note that VERTREG does not check for the presence of duplicate prior information names when it adds its new prior information equations to the existing PEST control file. Hence it is up to the user to ensure that the above naming strategy does not cause a name clash with prior information that may already be present within the existing PEST control file. (Of course you should always run PESTCHEK on any PEST control file written by VERTREG; it will soon inform you of nonuniqueness in prior information names if this has occurred.)

VERTREG next prompts:-

```
Do you require data-density-dependent weighting?  [y/n]:
```

As mentioned above, VERTREG provides two options for the assignment of weights to the prior information equations that it introduces to the PEST control file. A uniform weight can be assigned, or weights can be varied in accordance with the necessity to strengthen the role of regularisation information in regions where measurement information is scarce. VERTREG implements the latter method of weights assignment by increasing prior information weights with distance of pilot points cited in those prior information equations from the nearest measurement point(s). In order to implement this strategy VERTREG must know measurement point locations. Hence it prompts:-

```
Enter name of data coordinates file:
```

The format of this file must be the same as that of a bore coordinates file; see Part A of this manual for details. A bore coordinates file must have at least three columns of data, with bore identifiers comprising the first column, followed by eastings in the second column, followed by northings in the third column. After having read this file, and thus informing itself of the locations of measurement points, VERTREG issues the following set of prompts:-

```
Weights function is "a+b*sum_over_closest_n(r**c)" :-
    Enter a:
    Enter b:
    Enter n:
    Enter c:
    Enter maximum allowable weight:
    Enter minimum allowable weight:
```

This weights calculation strategy is now explained in greater detail.

Each new item of prior information which VERTREG introduces to the PEST control file cites two pilot points (strictly speaking, parameters based on two pilot points), one from each pilot points file. For each one of these points, VERTREG evaluates a trial weight, calculated as:-

$$w = a + b \sum_{i=1}^{n} r_i^c$$

where *a*, *b*, *n* and *c* are supplied in response to VERTREG's prompts above, and $r_i$ is the distance between the pilot point and the *i*'th closest measurement point. After calculating a notional weight for both of the pilot points involved in the prior information equation, VERTREG then takes the greater of these two weights. If this exceeds the maximum allowable weight, or is less than the minimum allowable weight (supplied by the user in response to the prompts shown above), VERTREG adjusts the calculated weight accordingly and assigns this to the prior information equation. At the time of writing this document it is strongly suggested that *n* be assigned the value of 1. Thus VERTREG calculates the prior information weight as a function of the distance between a pilot point and the closest measurement point to it. The greater is this distance, the greater is the calculated weight (unless *b* is zero and/or *c* is zero).

Alternatively, the user can instruct VERTREG that no density-dependent weighting is required. It will then prompt:-

```
Enter weight for all new prior information:
```

in response to which a weight greater than or equal to zero should be supplied.

Finally VERTREG prompts:-

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

The filename which you provide should include an extension of "*.pst*", as is the custom for PEST control files.

Where there are many pilot points in each of the pilot points files supplied to VERTREG, construction of the set of new prior information equations may take a minute or two. When VERTREG has finished execution it informs the user that the new PEST control file has been written. It also lists any pilot-point-based parameters (pertaining to the nominated pilot point files) that are omitted from the new prior information equations by virtue of the fact that the distance between each such point and its closest neighbour in the other layer is greater than the user-supplied exclusion distance. If points are omitted which should not be omitted, the user may then re-run VERTREG with the exclusion distance set higher.

**Uses of VERTREG**

VERTREG is used in the construction of a PEST control file required for mutli-layer regularised inversion where spatial parameterisation is based on pilot points. Normally it will be run as part of a sequence of programs used to add information to this PEST control file; other programs in the sequence will probably include PPK2FAC and PPKREG. For a model comprised of many layers, PPKREG may need to be run many times in order to add prior information equations encapsulating horizontal regularisation linkages for all model layers; VERTREG may then need to be run just as many times in order to incorporate vertical regularisation linkages. The latter will most often be required where horizontal regularisation is of the "smoothness" type. In this case, as was explained above, the introduction of vertical regularisation promulgates the existence of a unique solution to the inverse problem that can exist (or "almost exist") even in the absence of field data. This brings stability to the inverse problem. The task of the regularised inversion process is then to calculate parameter values which deviate only just enough from this "preferred system condition" to allow a good fit between model outputs and field data to be obtained. The result is the estimation of a set of parameter values that, provided the "preferred condition" is well chosen, "make hydrogeological sense".

It should be noted in passing that when working in layered model domains, the use of inter-layer head *differences* as part of the calibration dataset can result in much better estimates of vertical conductivities/conductances than would otherwise be possible. See documentation of program LAYDIFF.

**See Also**

See also FAC2REAL, LAYDIFF, PPK2FAC and PPKREG

# ZONE2BLN

**Function of ZONE2BLN**

Program ZONE2BLN writes a SURFER "blanking" file representing polygons defining the borders of various zones within the finite-difference grid. Zones are collections of cells of constant value as defined in a user-supplied MODFLOW/MT3D-compatible integer array.

**Using ZONE2BLN**

A settings file `settings.fig` must be present in the directory from which ZONE2BLN is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution ZONE2BLN prompts:

```
 Enter the name of the grid specification file:
```

Type in the name of the grid specification file. Alternatively, if the filename file `files.fig` is present in your current directory, ZONE2BLN may provide a default filename (as read from the filename file) with the above prompt; press <Enter> to accept the default or type in the desired name. ZONE2BLN uses the grid specification file to obtain the geographical information it needs in order to represent grid zonation in a blanking file.

Next ZONE2BLN prompts:

```
 Enter name of zoned integer array file:
```

Here supply the name of a formatted or unformatted file holding a MODFLOW/MT3D-compatible integer array (see Section 2.10 of Part A of this manual.) Zonation within this array is expressed as collections of cells of identical value. It is the role of ZONE2BLN to draw the outer boundary of each such zone represented in the file. However no outer boundary is drawn around the zone defined by cells whose value is zero in the integer array unless these cells are completely surrounded by cells of non-zero value.

ZONE2BLN's final prompt is:

```
 Enter name for SURFER blanking output file:
```

Here supply a filename of your choice. However it is recommended that it include an extension of "`BLN`" so that SURFER is able to recognise the file as a blanking file.

A blanking file written by ZONE2BLN can be used either for drawing the grid zonation, or for blanking those parts of a SURFER contouring grid that lie outside the active part of the finite difference grid. If used in the latter manner, an integer array should be supplied in which only two zones are represented, viz. the active and

inactive parts of the model grid, the former defined by cells of uniform non-zero value and the latter by zero-valued cells. If more than one zone is defined, the operation of the SURFER blanking function may have undesirable consequences as SURFER successively blanks the exteriors of all the different zones represented in the blanking file.

As written by ZONE2BLN the header to the section of the blanking file defining each zone contains two numbers, the first being the number of points representing the zone boundary, the second informing SURFER whether to blank inside or outside of the respective zone. ZONE2BLN provides a value of zero for this second number, informing SURFER to blank outside the zone. If this is not suitable for your purposes (for example if there is an inactive "hole" within the finite-difference grid), the number can be altered using a text editor. This second number has no effect if the blanking file is simply used for drawing purposes.

**Uses of ZONE2BLN**

The uses of ZONE2BLN in generating a contour map in which areas outside the study area (as defined by the active region of the model grid) are blanked has already been discussed. However SURFER can use blanking files for more purposes than simply blanking out regions, for blanking files represent one option for the importation of map data into SURFER. A striking effect can be achieved if a map of parameter zonation within the finite difference grid is overlain on a map of the grid itself (drawn from a file produced by GRID2DXF or GRID2BLN), with the zonation boundaries drawn thicker than the grid cell boundaries. The figure below shows such a picture. Note that an integer array representing parameter zonation can be produced from a real array containing model parameters with the help of program REAL2INT.



**Parameter zonation within a finite difference grid.**

**See Also**

See also GRID2BLN, GRID2DXF, ZONE2DXF.

# ZONE2DXF

**Function of ZONE2DXF**

Program ZONE2DXF writes a DXF file defining the borders of various zones within the finite-difference grid. Zones are collections of cells of constant value as supplied in a MODFLOW/MT3D-compatible integer array. The DXF file produced by ZONE2DXF can be used by plotting, contouring and GIS software to produce a map of model grid zonation.

**Using ZONE2DXF**

A settings file `settings.fig` must be present in the directory from which ZONE2DXF is run. Among other things, this file specifies whether a "number of columns, number of rows" header is used in formatted integer and real array files.

On commencement of execution ZONE2DXF prompts:

```
 Enter name of grid specification file:
```

Type in the name of a grid specification file. Alternatively, if a filename file (`files.fig`) is present in your current directory, ZONE2BLN may include a default filename (as read from the filename file) with the above prompt. In this case either press <Enter> to accept the default or type in the desired name. ZONE2DXF uses the grid specification file to obtain the geographical information it needs in order to produce a DXF file representing grid zonation.

Next ZONE2DXF prompts:

```
 Enter name of zoned integer array file:
```

Here supply the name of a formatted or unformatted file holding a MODFLOW/MT3D-compatible integer array (see Section 2.10 of Part A of this manual). Zonation within this array is expressed as collections of cells of identical value; it is the role of ZONE2DXF to draw the boundary of each such zone. However no outer boundary is drawn around the zone defined by cells whose value is zero unless such cells are completely surrounded by cells of non-zero value. (ZONE2DXF assumes that zero-valued cells are inactive.)

Finally ZONE2DXF prompts:

```
 Enter name for DXF output file:
```

Enter a filename of your choice; however it is recommended that you give the file an extension of "DXF" in order that it be recognised as a DXF file by the plotting, contouring and GIS software which may use it.

**Uses of ZONE2DXF**

The primary role of ZONE2DXF is to provide a mechanism whereby grid zonation can be displayed in GIS, plotting and contouring software. A user can define grid zonation within any model preprocessor that allows the construction and exporting of a MODFLOW-compatible integer array. Note also that an integer array representing parameter zonation can be produced from a real array containing model parameters with the help of program REAL2INT.

A striking effect in black-and-white can be achieved if a map of parameter zonation within the finite difference grid is overlain on a map of the grid itself (drawn from a file produced by GRID2DXF), with the zonation boundaries drawn thicker than the grid cell boundaries. The figure below shows such a picture.

**Parameter zonation within a finite difference grid.**

**See Also**

See also GRID2BLN, GRID2DXF, ZONE2BLN.