Groundwater Data Utilities

Part B: Program Descriptions

(See Part C for a description of programs pertaining to unstructured grid models.)

Watermark Numerical Computing

January, 2024

PREFACE

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

See also part C of the manual to the Groundwater Data Utilities. This contains programs written in support of unstructured grid models. The PLPROC program (that has its own separate manual) provides further support for unstructured grid models.

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.

I wish to gratefully acknowledge support that was provided to me for the writing of many of the utilities documented herein. In particular:-

- The original set of 30 utilities was developed as part of a project funded by the Australian Land and Water Resources Research and Development Corporation (LWRRDC) and by the Queensland Department of Natural Resources (QDNR).

- GENREG and PARM3D were developed under contract from the Madison Wisconsin office of USGS.

- RSM2SRF, RDAT2TAB, FAC2RSM and PPK2FACR were developed under contract from the South Florida Water Management District.

- Other utilities (including some of the unstructured grid utilities) were written with support from the National Centre for Groundwater Research and Training, Flinders University, Australia.

I wish to also acknowledge continued assistance provided by S. S. Papadopoulos and Associates. Assistance from EMS-I is also gratefully acknowledged and appreciated.

John Doherty

ALPHABETICAL LISTING OF GROUNDWATER DATA UTILITIES
(See Part C of the manual for the Groundwater Data Utilities for listing and documentation of unstructured grid utilities.)

adjobs	Adjusts observation weights for different observation groups in a PEST control file according to user-defined formulae.
anis2vtk	Creates a VTK file through which axes of anisotropy can be visualized at many points. A VTK file can be imported into PARAVIEW for 3D visualisation.
arr2bore	Undertakes spatial interpolation from a single array to a set of points.
arrayobs	Facilitates the introduction of model outputs comprised of MODFLOW/MT3D-compatible real arrays into a PEST parameter estimation process.
arrdet	Lists the contents of a MODFLOW or MT3D unformatted head/drawdown/concentration output file.
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.
bud2smp1	Similar to BUD2SMP, but easier to use where a model has a large number of layers.
conc2elev	Computes the elevation of the freshwater/saltwater interface on the basis of a sequence of concentration arrays.
dar2smp	Translates system states computed by a FEFLOW model to bore sample file format.
dbl2sgl	Re-writes a binary, double precision, MODFLOW-generated, head/drawdown file as a single precision binary file.
elev2conc	Computes a sequence of initial concentration arrays (one for each model layer) based on a user-supplied freshwater/saltwater interface elevation array, and (spatially varying) thickness of the interface.
elev2conc1	Similar to elev2conc, but computes “zero flow head” arrays as well.
fac2fefl	Uses PPKFAC_FEFL-generated kriging factors to modify a FEFLOW model input data file on the basis of spatial interpolation from a set of pilot points.
fac2fem	Uses ppk2fac-generated kriging factors to produce a MicroFEM input file on the basis of spatial interpolation from a set of pilot points.
fac2g	Complements PPK2FACG. Performs interpolation to a set of points, recording interpolated values in a single column file.

fac2real	Uses PPKFAC-generated kriging factors to produce a MODFLOW-compatible real array on the basis of spatial interpolation from a set of pilot points.
fac2real3d	Uses PPK2FAC3D-generated kriging factors to produce a set of MODFLOW-compatible real arrays through spatial interpolation from a set of three-dimensional pilot points.
fac2rsm	Uses PPKFACR-generated kriging factors to produce an RSM model input data file 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.
fieldgen_sva	Generates stochastic fields for a single layer of a structured grid model. Hyper-parameters that govern stochasticity can vary throughout the model domain; stationarity is not assumed. ("SVA" stands for "spatially varying anisotropy.)
fieldgen2d_sva	Generates a 2D stochastic field for a structured grid or unstructured grid model. Hyper-parameters that govern stochasticity can vary throughout the model domain; stationarity is not assumed.
fieldgen2d_sva1	Similar to FIELDGEN2D_SVA. However the grid over which spatial integration takes place does not need to be the same as that which hosts stochastic fields.
fieldgen3d_sva	Generates a 3D stochastic field for a multi-layer structured grid or unstructured grid model. Hyper-parameters that govern stochasticity can vary throughout the model domain; stationarity is not assumed.
fieldgen3d_sva1	Similar to FIELDGEN3D_SVA. However the grid over which spatial integration takes place does not need to be the same as that which hosts stochastic fields.
fieldgen3d_sva2	Similar to FIELDGEN3D_SVA1. However undertakes spatial averaging over a user-supplied set of independent normal variates. Suitable for parameter estimation.
genreal2srf	Interpolates from a MODFLOW grid of arbitrary specifications to the nodes of a SURFER grid file.
genreg	Inserts prior information pertaining to many different types of regularisation into an existing PEST control file.
getmularr	Extracts arrays from MODFLOW/MT3D unformatted output files at user-nominated simulation times and stores these arrays in separate formatted files.

getmularr1	Extracts all arrays for a nominated simulation time from a MODFLOW/MT3D unformatted output file and writes these to another unformatted MODFLOW/MT3D output 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.
hanis2bln	Creates a SURFER BLN file showing anisotropy axes at different points.
iidgen	Generates random numbers for use by programs FIELDGEN2D_SVA1 and FIELDGEN3D_SVA1.
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.
logarray	Evaluates the log (to base 10) of all elements of a real array.
many2one	Splits MODFLOW/MT3D-generated unformatted files comprised of multiple two-dimensional results arrays into individual formatted/unformatted files.
mf2vtk	Writes a “legacy VTK” file through which a MODFLOW grid, properties and calculated system states can be viewed using software such as PARAVIEW.
mf2vtk1	Similar to MF2VTK. However it reads MODFLOW input data from one or a number of tabular data files and provides default values for missing cells.
mkppstat	Writes a pilot point statistical specification file for the use of PPCOV_SVA in which local variogram “a” values reflect local pilot point density.
mkppstat3d	Writes a pilot point statistical specification file for the use of PPCOV3D_SVA in which local variogram “a_hmax”, “a_hmin” and “a_vert” values reflect local pilot point density.

mod2array	Reads a MODFLOW or MT3D input file, extracting real or integer arrays from that file and storing them in separate 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.
mod2obs_dbl	Same as MOD2OBS. However it reads a double precision MODFLOW system state output file (such as is generated by a MODFLOW 6 DIS model).
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.
mod2smpdiff	Interpolates the information contained in an unformatted MODFLOW/MT3D output file to user-specified bores, calculating the difference or ratio between heads/concentrations at user-nominated pairs of bores.
parcov	Builds a geostatistically-based covariance matrix for a set of parameters whose coordinates are provided.
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.
pestprep1	Similar to PESTPREP. However provides extra flexibility in observation naming.
pestprep2	Similar to PESTPREP1. However allows extra observation data to be added to an existing PEST input dataset.
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.
ppcov3d	Builds a covariance matrix pertaining to three dimensional pilot point parameters based on one or a number of geostatistical structures.
ppcov_sva	Builds a covariance matrix pertaining to two-dimensional pilot point parameters under the assumption that anisotropy, and other geostatistical properties, can vary spatially.

ppcov3d_sva	Builds a covariance matrix pertaining to three-dimensional pilot point parameters under the assumption that anisotropy, and other geostatistical properties, can vary spatially.
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.
ppk2facg	Calculates kriging factors for interpolation from a set of unnamed points where the coordinates and zone numbers of the latter are arranged in three vertical columns.
ppk2fac1	Identical to ppk2fac except for the fact that the regularisation data file which it writes is suitable for the use of ppkreg1.
ppk2fac2	Identical to ppk2fac1 except for the fact that it prompts for a blanking radius.
ppk2fac3	Contains improvements to ppk2fac[1-3] which allow it to work better in thin alluvial model domains, especially where complex tributary systems exist.
ppk2fac3d	Calculates kriging factors for interpolation from a set of three-dimensional pilot points to a series of MODFLOW-compatible real arrays.
ppk2facr	Calculates kriging factors for use in spatial interpolation from a set of pilot points to the nodes of an RSM mesh. Regularisation data file protocol is identical to that of PPK2FAC1.
ppk2fac_fefl	Calculates kriging factors for use in spatial interpolation from a set of pilot points to the elements of a FEFLOW mesh. Regularisation data file protocol is identical to that of PPK2FAC1.
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”.
ppsamp	Used in calibration-controlled Monte Carlo analysis. Samples stochastic fields at pilot point locations, interpolates between the pilot points and generates difference fields.
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.
rdat2tab	Reads an RSM element data file or index file. Adds mesh centroid coordinates to respective data elements and re-writes data in tabular format.
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.
rsm2srf	Reads an RSM (also GMS) 2d mesh file. Writes files through which SURFER can plot mesh design, mesh outer boundary, as well as nodes and element centroids.
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.
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.

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.
smpdiff	Writes a new bore sample file in which differences are taken between successive values in an existing bore sample file, or between values in an existing file and a reference value.
smpstat	Reads a bore sample file. Writes a text file containing some statistics for data associated with each bore.
smptrend	Writes a new bore sample file in which differences are taken between samples within an existing bore sample file and either the first sample for each bore in that file or a reference sample. However sampling is restricted to a yearly sample window.
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.

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.

ANIS2VTK

Function of ANIS2VTK

ANIS2VTK creates a VTK file composed of linear segments. Three lines are associated with each of a series of points in 3D space. These lines point in the directions of the three axes of spatial correlation anisotropy. Their lengths are proportional to the correlation length in each of these three orthogonal directions.

Using ANIS2VTK

Commence execution of ANIS2VTK by typing its name in a command line window. Its first prompt is:

```
Enter name of anisotropy specification file:
```

The specifications of an "anisotropy specification file" are flexible. However certain rules apply. It can be a tabular data file or a CSV file. It must contain at least 9 columns. Rules are as follows:

- Columns which contain *x*, *y* and *z* coordinates must follow each other;
- These must precede columns that contain the following (in the order shown below):
    - *ahmax*
    - *ahmin*
    - *avert*
    - *bearing*
    - *dip*
    - *rake*

ahmax, *ahmin* and *avert* are correlation lengths (or "*a*" values of pertinent variogram equations) in each of three perpendicular directions. Often (but not always) the direction of *avert* approximates the vertical direction. Hence it is normally the smallest of these three correlation lengths.

The last three of the above six variogram hyperparameters (i.e. *bearing*, *dip* and *rake*) denote the directions of these three perpendicular axes.

Bearing must be between -360 degrees and 360 degrees. This is the direction in which the positive direction of the *a_hmax* axis points. It is measured counterclockwise from north. *Dip* is the vertical angle in which the positive direction of the *a_hmax* axis points. Values for *dip* can vary between -180 degrees and 180 degrees. Note that, to conform with GSLIB protocols, a downwards dip is negative and an upwards dip is positive. The last variable is *rake*. Picture yourself looking down the *a_hmax* axis from its positive end towards its negative end. Now picture the ellipsoid of anisotropy being rotated anticlockwise about this *a_hmax* axis so that the positive direction of the

a_hmin axis rises. The angle by which it rises is the *rake*. The value supplied for *rake* must be between -90 degrees and 90 degrees.

If you like, the anisotropy specification file can have more than 9 columns. Furthermore, the six juxtaposed *ahmax*, *ahmin*, *avert*, *bearing*, *dip* and *rake* columns do not need to immediately follow the three juxtaposed *x*, *y* and *z* columns; these sets of columns can be separated from each other by columns that contain other variables.

ANIS2VTK asks where the above two sets of columns begin:

```
In what column do point coordinates begin?
In what column do variogram hyperparameters begin?
```

ANIS2VTK allows you to read a further four columns of data, and to provide labels for these data. (These columns can contain coordinates or hyperparameter data if you wish.) This allows you to colour the lines after you import the VTK file into a platform such as PARAVIEW.

ANIS2VTK asks:

```
Read data from column number (<Enter> if none):
Provide a label for this data:
```

ANIS2VTK next asks whether the file that it must read contains a header:

```
Skip a line at top of file? (y/n):
```

Next it asks for the name of the VTK file that it must write.

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

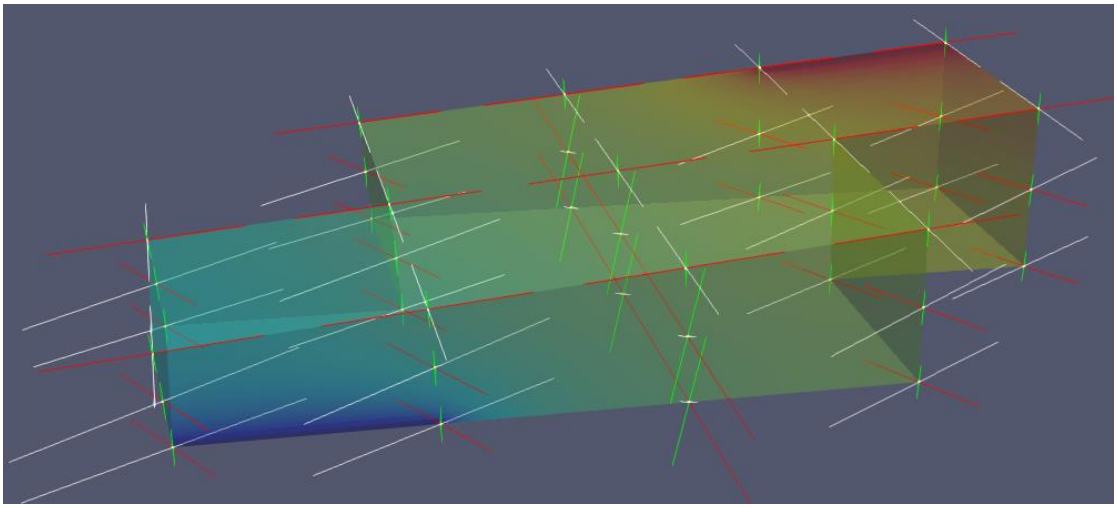
Finally, it asks what factor to apply to correlation lengths. If this factor is 1.0 then the lengths of the lines that are centred on each conceptual point are equal to two times *ahmax*, *ahmin* and *avert*. (They extend each of these distances in each direction from the point.) See below.

Using ANIS2VTK

As was stated above, ANIS2VTK allows you to associate up to four types of data with lines that are specified in its output file. It associates one item of data with each of these lines itself. The label for this type of data is “atype”. Lines that point in the direction of *ahmax* receive an atype value of 1; those that point in the direction of *ahmin* receive an atype value of 2; those that point in the direction of *avert* receive an atype value of 3. Thus three atypes are associated with each point (because 3 lines are associated with each point).

Uses of ANIS2VTK

Obviously, ANIS2VTK is used to make illustrative pictures.



See Also

See also LIN2VTK and HANIS2BLN.

ARR2BORE

Function of ARR2BORE

ARR2BORE undertakes spatial interpolation from a single MODFLOW or MT3D-compatible real array to a set of points whose coordinates are cited in a bore coordinates file. The array from which interpolation takes place can be housed in a formatted or unformatted file. As for the MOD2OBS and MOD2SMP utilities, interpolation from the array to the points is bilinear, with account taken of inactive cells. The latter are identified as cells for which absolute real-array values are above a user-specified threshold.

Using ARR2BORE

Like all programs of the Groundwater Data Utility suite, ARR2BORE checks for the presence of a file named *settings.fig* in the directory from which it is run. The contents of this file inform ARR2BORE of whether formatted real array files include a number-of-columns, number-of-rows header or not.

After having read *settings.fig* (or complained of its absence), ARR2BORE asks for the name of the grid specification file pertaining to the current model:-

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

in response to which the name of this file should be supplied. Once it has read this file, ARR2BORE prompts for the name of a bore coordinates file, and then a bore listing file:-

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

(These can be the same file if desired). The format of these files is described in Part A of this manual. The former contains three columns of data, with the first column listing bore names (which should be 20 characters or less in length), and the second and third listing bore eastings and northings. If desired, the fourth (layer number) column can be omitted, for ARR2BORE does not read this column.

The bore listing file comprises a selection mechanism for points to which spatial interpolation is required. This should contain a single column in which bore names are listed; if more than one column is present within this file, columns after the first are simply ignored. The bores listed within this file must be a subset of bores appearing in the bore coordinates file; only points listed in the bore listing file will appear in ARR2BORE's output file, for it is only to these points that spatial interpolation takes place.

ARR2BORE next asks for the name of the real array file which it must read. This can be a formatted or unformatted file, with extensions of "*ref*" and "*reu*" being the defaults for the former and latter file types respectively. ARR2BORE's prompt is:

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

Inactive elements of the array contained in this file are identified as those whose absolute values are above a certain threshold, this threshold being supplied by the user in response to the prompt:-

```
Enter inactive threshold for this array (<Enter> if 1E35):
```

The method of spatial interpolation employed by ARR2BORE is the same as that used by MOD2OBS and MOD2SMP. This is a bilinear scheme, with modifications for proximity to inactive cells. If a point to which interpolation must take place is outside of the grid, or within an inactive cell, this condition is stated in the ARR2BORE output file. If the interpolation point is located at a place for which one of the four cell centres that enclose it is inactive (or it is not in fact enclosed by four cell centres at all due to proximity to the edge of the grid), the bilinear interpolation scheme is adjusted to compensate for this missing data.

ARR2BORE's final prompt is:-

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

ARR2BORE writes its output data in tabular form, the output table being comprised of four data columns. The first of these columns contains bore identifiers. The second and third contain bore eastings and northings as read from the bore coordinates file. The fourth contains numbers obtained through spatial interpolation from the real array to the points listed in the bore listing file.

Uses of ARR2BORE

ARR2BORE is useful wherever spatial interpolation from a model array to a set of discrete points must take place. For example it can be used to undertake spatial interpolation from a freshwater/saltwater interface elevation array written by CONC2ELEV, to a set of observation wells.

See Also

See also MOD2OBS, MOD2SMP and CONC2ELEV.

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.

ARRDET

Function of ARRDET

"ARRDET" stands for "ARRay DETails". It reads an unformatted MODFLOW heads or drawdowns output file, or a MT3D unformatted concentrations output file, and lists to a user-specified ASCII file the contents of the headers to arrays found in this file.

Using ARRDET

ARRDET is run by typing its name at the command-line prompt. It obtains information from the user through the user's response to a series of its own prompts. As for other members of the Groundwater Data Utilities, if the user responds to any such prompt with the single letter "e", operation of ARRDET will return to its previous prompt.

Like most programs of the Groundwater Data Utility suite, ARRDET begins execution by prompting for the name of the grid specification file pertaining to the current model. The prompt is:-

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

If a filename file (named *files.fig*) is present in the current working directory, and if this file lists the name of the grid specification file for the current model, this name will be included in the above prompt. To accept this filename simply press the <Enter> key; otherwise, provide ARRDET with the name of the correct grid specification file.

Next ARRDET asks for the name of the MODFLOW/MT3D unformatted output file which it must read:-

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

and whether this is a MODFLOW or MT3D output file (the array headers for these two file types are different).

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

Respond with "f" or "t" as appropriate.

Next ARRDET prompts for the name of the output text file which it must write:-

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

It then reads the unformatted MODFLOW/MT3D output file, listing the contents of array headers found therein to its own output file. The user can thus become aware of the contents of such files (which are otherwise hidden because of their binary nature).

Uses of ARRDET

ARRDET is useful as a precursor to the use of GETMULARR. An altered ARRDET output file can serve as a GETMULARR input file.

See Also

See also GETMULARR, GETMULARR1, MOD2OBS, MOD2SMP, MANY2ONE

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 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 MODFLOW 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 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, BUD2HYD 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 BUD2HYD 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.

BUD2HYD'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 20 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.

BUD2SMP1

BUD2SMP1 is identical to BUD2SMP except for the fact that it reads a single integer array pertinent to all layers rather than a different integer array for each layer. It is thus easier to use with models that have a large number of layers.

In all other respects its use and functionality are identical to that of BUD2SMP.

CONC2ELEV

Function of CONC2ELEV

CONC2ELEV was built primarily for use with the SEAWAT model (which is an amalgam of MODFLOW and MT3D). It performs the converse operation to the ELEV2CONC utility. It reads a set of concentration arrays written by SEAWAT (that is, the MT3D component of SEAWAT) and computes the elevation of the freshwater/saltwater interface, this being identified according to a user-specified concentration threshold (for example a concentration half way between that of fresh water and salt water). It writes the interface elevation to a MODFLOW-compatible real array. It writes some further information pertaining to the location of this interface to another file for use by other model post-processing software.

Using CONC2ELEV

Execution of CONC2ELEV is commenced by typing its name at the screen prompt. It then asks the user a series of questions. If, in response to any of these questions, the response is simply "e" followed by the <Enter> key, execution of CONC2ELEV returns to its previous prompt. Thus recovery from mistaken input is a simple matter.

Upon commencement of execution CONC2ELEV looks for a file named *settings.fig* in the directory from which its execution was initiated. As explained in Part A of this manual, this file informs members of the Groundwater Data Utilities of the necessity (or otherwise) for inclusion of a number-of-columns, number-of-rows header in real and integer array files which they read/write. It also informs them of which date format to employ (this being ignored by CONC2ELEV as this program does not read or write dates.)

Like most programs of the Groundwater Data Utility suite, CONC2ELEV begins execution with the prompt:-

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

If a "filenames file" named *files.fig* is present in the directory from which CONC2ELEV is run, the name of a default grid specifications file may be included in the above prompt. This can be accepted simply through pressing the <Enter> key in response to the above prompt.

Next CONC2ELEV asks:-

```
Enter upper layer for processing:
Enter lower layer for processing:
```

Reply to each of these prompts with an integer between and including 1 and the number of layers in the model, with the second integer greater than the first. CONC2ELEV will only search for the freshwater/saltwater interface between the nominated layers. (This can be useful where different interfaces exist in different geological units separated by an aquitard; if there are indeed multiple interfaces

within the sequence of layers in which CONC2ELEV undertakes its search, then it will only provide the elevation of the highest of these interfaces.)

CONC2ELEV's next prompt is:

```
Enter filename base for layer Nu to Nl bottom elev arrays:
```

In this prompt N_u is one less than the layer number of the upper model layer supplied above, while N_l is equal to the lower layer number supplied above. In responding to this prompt it is assumed that the bottom elevation of the layer above the uppermost layer of interest is equal to the elevation of the top of that layer. Where the uppermost layer of interest is layer 1, N_u is 0.

Suppose that the user responds to the above prompt with the string "*bottom*". Then CONC2ELEV looks to files *bottomN.ref* where *N* ranges from N_u to N_l for these arrays. Thus if the search for the freshwater/saltwater interface is to take place over layers 1 to 15, CONC2ELEV will read files *bottom0.ref, bottom1.ref* ….*bottom15.ref*. These arrays can be easily extracted from a MODFLOW discretisation file using the MOD2ARRAY utility described elsewhere in this manual. (As is the usual convention, a number-of-columns, number-of-rows header is expected in each of these arrays if the COLROW variable in file *settings.fig* is set to "yes".)

CONC2ELEV reads concentration arrays for all layers of interest from an MT3D unformatted output file (i.e. a "UCN file"). It next prompts for the name of this file, and the simulation time for which these arrays must be read.

```
Enter name of unformatted MT3D concentration file:
Enter simulation time to read arrays for:
```

It then asks:-

```
Enter threshold concentration defining interface:
Enter threshold concentration defining inactive cell:
```

When computing the elevation of the interface in any vertical column of the finite difference grid, CONC2ELEV starts at the highest active cell and works downwards. It recognises active cells as those for which the absolute value of concentration is below a certain threshold. Obviously this threshold must be higher than any concentration likely to be computed by the model; the user should keep this in mind when assigning a value to the MT3D CINACT variable. Setting this variable to 1E30 is normally a safe option.

In searching downwards through a particular vertical column of the model, CONC2ELEV records the location at which the concentration increases above a user-supplied threshold as the location of the interface. If this threshold is crossed between cell centres, CONC2ELEV estimates the elevation of the crossing of this threshold through linear interpolation between cell elevation midpoint elevations. A problem arises however if, for the first active cell encountered, the concentration is already above the user-specified interface threshold concentration. In this case CONC2ELEV will either provide an interface elevation that is equal to that of the midpoint of the highest active cell, or will provide a dummy elevation value, its choice in this matter being governed by the user's response to the following prompt:-

```
Use dummy value or cell midpoint elev when interface above/below top/bottom
  active cell centre? [d/m]:
```

If the "d" option is selected, CONC2ELEV then asks:-

```
Enter dummy value for above top cell centre:
Enter dummy value for below bottom cell centre:
```

in response to which appropriate values should be supplied. An advantage of the "cell midpoint elevation" option is that the interface elevation will not show discontinuous changes in any cell if, in response to a parameter change, the interface elevation rises higher than the highest active cell centre or drops below the lowest active cell centre as parameters are changed; this is useful if the interface elevation at certain locations is part of a calibration dataset used by PEST for estimating model parameters.

After it has computed the elevation of the interface in every vertical column in which at least one active cell exists, CONC2ELEV prompts for the name of its output file. The prompt is:-

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

This is a real array like any other (which can be formatted or unformatted, and will contain a number-of-columns, number-of-rows header or not, depending on the COLROW setting in file *settings.fig*). A dummy value of 1E35 is assigned to any element of this array for which no cell within the corresponding column of the finite difference grid is active (and for which computed concentration information is therefore unavailable).

After having written this file CONC2ELEV asks:-

```
Enter name for row/column intersection file:
```

The "row/column intersection file" provides information on the location of the interface along each row and column of the finite difference grid within each model layer. Part of such a file is shown below.

```
Row intersections of interface for layer 3 ---->
      1      1    1         4848.029         5223.517         4380.713
      2      1    1         4847.601         5273.146         4293.896
      3      1    1         4846.760         5322.417         4206.874
      4      1    1         4845.543         5371.363         4119.663
      5      1    1         4844.005         5420.031         4032.291
      6      1    1         4842.220         5468.486         3944.796
      7      1    1         4840.262         5516.790         3857.214
      8      1    1         4838.201         5565.005         3769.582
      9      1    1         4836.156         5613.234         3681.957
     10      1    1         4834.279         5661.608         3594.415
     11      1    1         4832.745         5710.280         3507.046
     12      1    1         4831.732         5759.403         3419.937
     13      1    1         4831.376         5809.095         3333.156
     14      1    1         4831.732         5859.402         3246.732
     15      1    1         4832.745         5910.280         3160.636
     16      1    1         4834.278         5961.608         3074.800
etc
```

Part of a row/column intersection file.

The row/column intersection file is divided into two segments per layer. Row intersections are provided in one of these segments while column intersections are provided in the other.

Within each segment there are as many entries as there are rows or columns within the model grid. The first entry on each line is the row or column number. Then follows

the number of interface intersections along that row or column. Suppose that this number is *N*; then *N* groups of 5 entries follow. The first member of each of these subgroups of 5 is an integer, this depicting the "intersection type"; this is "1" if concentration rises across the interface and "-1" if concentration falls across the interface (in the direction of increasing column or row index along each row or column respectively). Then follows the distance of the interface from either the left margin of the grid (in the case of rows) or the top of the grid (in the case of columns). Following that are the easting, northing and elevation of the interface along the row or column centre line. The location of the interface along a row or column centre line is determined as that point where the interface concentration threshold is encountered; where this is not encountered exactly at a cell centre, the location is determined through horizontal linear interpolation between cell centres. Because computed cell concentrations are employed in performing these calculations, and because concentrations are notionally assigned to the centre of each cell, interface elevations recorded in the row/column interface file effectively pertain to layer vertical midpoints (the locations of which may not be constant along each row or column of the grid). Where the freshwater/saltwater interface occurs between cell centres, layer vertical midpoint elevation is obtained through linear interpolation of cell midpoint elevations.

There is one idiosyncrasy of CONC2ELEV's interface computation algorithm of which the user should be aware. If active cells are underlain by inactive cells, to be underlain by active cells again, and if the concentrations in upper level active cells are all below the interface threshold concentration while those in the lower cells are all above the interface threshold concentration, CONC2ELEV will assign the interface elevation a value equal to the elevation of the midpoint of the uppermost of the lower group of cells, or the upper dummy concentration as specified by the user (according to the option selected by the user above). This unusual situation can, under some circumstances, lead to discontinuities in interface elevation with changes in parameters if these changes cause the interface to encroach into the upper group of layers. However this situation is considered to be atypical.

Uses of CONC2ELEV

The elevation of the freshwater/saltwater interface is a somewhat abstract notion as concentrations change gradually with depth; furthermore complex groundwater flow patterns take place in the vicinity of this interface. Nevertheless, this quantity is often provided as an outcome of measurements taken in monitoring wells, and hence has the potential to be useful in both the model calibration process and in the assignment of initial concentrations to the model domain. CONC2ELEV can be run as part of a model calibrated by PEST when interface elevations are used in the former capacity. CONC2ELEV's sister program ELEV2CONC can be used in the latter capacity.

See Also

See also ELEV2CONC, MOD2ARRAY.

DAR2SMP

Function of DAR2SMP

DAR2SMP translates FEFLOW outputs to bore sample file format. It is assumed that these outputs pertain to observation wells whose coordinates and other details have been provided to FEFLOW by the user. Once translated to this format, the following operations are easily implemented, either as part of the model run by PEST through the calibration process, or prior to the calibration process as part of PEST pre-processing:-

1. temporal interpolation of FEFLOW-calculated quantities to the times at which field-measured counterparts to these quantities were actually observed (see the SMP2SMP utility);

2. mathematical manipulation of these quantities, including digital filtering (see the TSPROC package belonging to the PEST Surface Water Modelling Utility suite);

3. automatic construction of all or part of a PEST input dataset (see the PESTPREP, PESTPREP1 and PESTPREP2 utilities).

DAR2SMP reads an ASCII FEFLOW-generated "DAR" file containing "reduced computational results". To write this file, FEFLOW must be run using a command such as:-

```
feflow -run -hide -work c:\feflow -steps tstep1.pow -dar output.dar new.fem
```

Run FEFLOW using the command:-

```
feflow -help
```

to see more FEFLOW usage details. Note that for both of the above commands, either the "feflow" string should be prefixed by the name of the directory containing the FEFLOW executable, or this directory should be including in the PATH environment variable.

Using DAR2SMP

DAR File Format

When building a FEFLOW model, the user can provide FEFLOW with the locations of wells in which field measurements have been made. In any particular study area these wells are, of course, designated according to local naming conventions. However in FEFLOW they are simply numbered. The first part of a DAR file lists these wells, together with their global and local coordinates. Part of a DAR file table in which this information is recorded is depicted below.

```
+++++++++++++++ FEFLOW Computational Results +++++++++++++
Problem file: new.fem
Thu May 17 08:57:53 2007
NT Rural Groundwater Modelling
Three Dimensions (3D)
*********************************************************************
LOCATION (GLOBAL AND LOCAL) OF OBSERVATION AND WELL POINTS:
---------------------------------------------------------------------
 Obs    x_g [m]       y_g [m]       x_l [m]       y_l [m]       z [m]
---------------------------------------------------------------------
  1 725009.800000 8613460.100000 27509.800000 20960.100000   -49.003606
  2 725729.800000 8614470.100000 28229.800000 21970.100000   -47.903440
  3 728149.800000 8614850.100000 30649.800000 22350.100000   -42.492337
  4 729849.800000 8615060.100000 32349.800000 22560.100000   -43.092606
  5 732589.800000 8615720.100000 35089.800000 23220.100000   -47.455392
  6 726989.800000 8613570.100000 29489.800000 21070.100000   -44.826290
  7 726979.800000 8612350.100000 29479.800000 19850.100000   -48.183831
  8 725544.800000 8616900.100000 28044.800000 24400.100000   -46.949874
  9 726979.800000 8615640.100000 29479.800000 23140.100000   -45.178760
 10 722929.800000 8622050.100000 25429.800000 29550.100000   -20.176454
...
  a 743629.700000 8631110.100000 46129.700000 38610.100000   -27.916524
  b 738231.000000 8625986.000000 40731.000000 33486.000000   -25.045553
  c 741129.700000 8623960.100000 43629.700000 31460.100000   -29.669442
  d 724289.800000 8623060.100000 26789.800000 30560.100000   -16.071621
  e 724169.800000 8622810.100000 26669.800000 30310.100000   -16.169266
  f 724399.800000 8622640.100000 26899.800000 30140.100000   -16.657809
  g 724679.000000 8622635.000000 27179.000000 30135.000000   -16.809188
...
_28 724649.800000 8621187.600000 27149.800000 28687.600000   -18.193765
_29 724859.800000 8621260.100000 27359.800000 28760.100000   -18.493536
_30 725112.880000 8621264.470000 27612.880000 28764.470000   -18.636139
```

Start of a FEFLOW-generated DAR file.

In a table such as the above, observation wells are indexed using positive integers starting at 1. Alphabetical indices, followed by integer indices with a leading underscore, pertain to entities at which water flows in or out of the model domain (such as wells and various boundary condition types); these are ignored by DAR2SMP.

The well location table that leads a DAR file is followed by a sequence of data tables, one pertaining to each FEFLOW time step, and/or to each time for which FEFLOW output is requested. Part of such a table appears below.

```
*********************************************************
RESULTS AT STEP =  14  AND TIME = 31.0000 [d]:
---------------------------------------------------------
 Obs     H [m]          Vx [m/d]       Vy [m/d]       Vz [m/d]
---------------------------------------------------------------
  1 2.209652e+001 -1.244975e-003 1.977176e-002 -3.782298e-005
  2 2.032368e+001 5.296040e-003  2.186053e-002 -1.050942e-004
  3 1.931409e+001 2.065794e-003  1.527837e-002 -5.245243e-005
  4 1.848264e+001 1.652906e-003  1.252860e-002  9.043694e-005
  5 1.863514e+001 -1.191209e-003 3.790130e-003 -5.312853e-005
  6 1.982329e+001 -8.428071e-003 3.890173e-003  2.724540e-003
  7 2.438048e+001 1.637838e-003  2.121529e-002 -6.158893e-005
  8 1.511694e+001 7.684201e-003  3.962595e-002 -1.738366e-005
  9 1.771841e+001 5.037181e-004  2.848033e-002 -1.012546e-004
 10 6.436490e+000 -1.158686e-002 8.368673e-002  7.140338e-005
  a 6.604564e+000 (single well: 7.218000e-001 [m3/d])
  b 1.044103e+001 (single well: 0.000000e+000 [m3/d])
  c 5.317448e+000 (single well: 0.000000e+000 [m3/d])
  d 5.339259e+000 (single well: 0.000000e+000 [m3/d])
  e 5.614646e+000 (single well: 0.000000e+000 [m3/d])
  f 5.806405e+000 (single well: 0.000000e+000 [m3/d])
  g 5.812766e+000 (single well: 0.000000e+000 [m3/d])
_28 7.676518e+000 (single well: 7.218000e-001 [m3/d])
_29 7.566448e+000 (single well: 0.000000e+000 [m3/d])
_30 7.531137e+000 (single well: 7.218000e-001 [m3/d])
```

Part of the output dataset pertaining to a particular FEFLOW time step.

In the above example only four quantities computed by FEFLOW for each observation well are listed at each time step. In other cases other quantities may also be listed (for example FEFLOW-calculated concentrations). Any of these can be read by DAR2SMP (but only one of these on any particular DAR2SMP run). As is documented below, the column to be read by DAR2SMP is specified by its header; in the above case there are four headers, namely “H”, “Vx”, “Vy” and “Vz”. (DAR2SMP ignores the units that accompany each header.)

Running DAR2SMP

DAR2SMP is run by typing its name at the command prompt. As for other members of the Groundwater Data Utility suite, information must be supplied by the user in response to questions posed by DAR2SMP. If the response to any such question is simply “e” or “E” followed by <Enter>, DAR2SMP will backtrack to its previous question, whereupon any mistakes made by the user in answering that question can be rectified.

Immediately upon commencement of execution, DAR2SMP checks for the presence of a “settings file” named *settings.fig* in the directory from which it is run. As is described in Part A of this manual, this informs DAR2SMP (and other utilities) whether to use the “dd/mm/yyyy” or “mm/dd/yyyy” convention in specifying dates. Optionally, a settings file also informs any program which reads it whether a “number of columns, number of rows” header is expected in MODFLOW-compatible integer and real arrays; this setting has no relevance in the FEFLOW context and is therefore not required in a settings file read by DAR2SMP. A typical *settings.fig* file is shown below.

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

A typical *settings.fig* file.

Once it has read the settings file, DAR2SMP asks:-

```
Enter name of FEFLOW DAR file:
```

in response to which the name of the DAR file to be processed by DAR2SMP should be supplied (surrounded by quotes if it contains spaces). Next DAR2SMP asks for the type of data which it should read from each time-specific output table. The prompt is:

```
Enter header for data type to extract from this file:
```

Respond with the appropriate header. In responding to this prompt, note the following.

1. Do not include the time units which accompany each header.

2. You do not need to match the case of header character(s) to those provided in DAR file output tables.

Next DAR2SMP asks for the name of an "observation number to bore identifier conversion" file. The prompt is:-

```
Enter name of obs number to boreid conversion file:
```

This is a file which relates observation well numbers as employed by FEFLOW to user-specified bore identifiers. Part of such a file is depicted in the following figure.

```
# FEFLOW_ID   DATABASE_ID
   1           BH100435
   2           BH245822
  17           BH173452B
  30           BH305544
  40           BH406788
  50           BH503453
 164           BH164567
```

Part of an "observation number to bore identifier" file.

The "observation number to bore identifier file" is easily prepared with a text editor. It should contain two columns, the first being observation well numbers as listed by FEFLOW, the second being bore identifiers as they are known by the user. The following aspects of this file should be noted.

1. Blank lines, and lines beginning with a "#" character (such as the optional header in the above example) are ignored by DAR2SMP.

2. A bore identifier must be 20 characters or less in length so that it conforms with bore sample file protocol.

3. There is no need for FEFLOW observation well numbers (first column in the "observation number to bore identifier" file) to be provided in increasing order.

4. A non-integer or negative integer FEFLOW observation well number is not allowed.

5. There is no need for all FEFLOW observation well numbers featured in a DAR file to be listed in an "observation number to bore identifier" file. Observation well numbers which are not associated with a bore identifier are simply ignored by DAR2SMP.

6. If a FEFLOW observation well number provided in an "observation number to bore identifier" file is not represented in a FEFLOW DAR file, it is simply ignored by DAR2SMP.

DAR2SMP next asks:-

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

(Note that the date protocol presented in the first of the above prompts will depend on that provided in file *settings.fig*.) DAR2SMP needs the above information so that it can convert elapsed simulation times as provided in DAR file data tables to dates and times as required in a bore sample file.

Finally DAR2SMP asks:-

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

Once you have provided this name, DAR2SMP writes the bore sample file as requested. An entry is provided in this file for each bore listed in the "observation to bore identifier" file, for each output time appearing in the FEFLOW DAR file. Use of the SMPCHEK utility will readily verify that the DAR2SMP output file is a valid bore sample file.

Uses of DAR2SMP

DAR2SMP comprises part of the FEFLOW PEST interface, other members of which are PPK2FAC_FEFL and FAC2FEFL.

As stated above, once FEFLOW data has been re-written in bore sample file format it is amenable to further processing by other members of the Ground and Surface Water Utility suites. For example the SMP2SMP utility can be employed for temporal interpolation to the times at which observations were made in the field; both DAR2SMP and SMP2SMP should then be run in sequence behind FEFLOW as part of the composite model calibrated by PEST. If this is done, automation of PEST input file construction for even a complex inversion problem then becomes a simple matter. See documentation of the PESTPREP, PESTPREP1 and PESTPREP2 utilities for more details.

See Also

See also PPK2FAC_FEFL, FAC2FEFL, SMP2SMP, PESTPREP, PESTPREP1 and PESTPREP2.

DBL2SGL

Function of DBL2SGL

DBL2SGL reads a double precision, binary, dependent-variable file written by MODFLOW or MT3D. For old versions of MODFLOW, this file can contain heads or drawdowns. For MT3D it contains concentrations. DBL2SGL can also read a double precision, binary file written by MODFLOW-USG and by MODFLOW 6 – but only if the model is of the structured grid (DIS) type in both of these cases. Model-calculated variables other than head and drawdown can be stored in files written by these latter programs.

Unless they were specifically compiled to do otherwise, MODFLOW, MT3D and MODFLOW-USG store calculated system states as single precision numbers. This does not apply to MODFLOW 6; it always stores calculated system states as double precision numbers. This means that many of the programs that are documented herein (programs such as MOD2OBS, MANY2ONE, ARRDET and others) cannot read binary files written by MODFLOW 6. To remove this problem, DBL2SGL can be used to read the file, and then rewrite its contents as single precision numbers so that utility programs documented herein can read it and process its contents.

Using DBL2SGL

Use of DBL2SGL is simple. It prompts for the name of the file that it must read and the name of the file that it must write. Its prompts are:

```
Enter name of double precision binary file:
Enter name for single precision binary file:
```

Uses of DBL2SGL

Because it translates a double precision system state file to a single precision system state file, DBL2SGL allows some of the MODFLOW/MT3D utility programs that are documented herein to be used with MODFLOW 6. It is important to note the following however:

- DBL2SGL can only read binary files that are generated by MODFLOW 6 for DIS (i.e. structured grid) models.

- Other utility programs are available that were written especially for use with MODFLOW 6. See part C of this manual.

Because DBL2SGL can rewrite a binary file generated by MODFLOW 6 as one which can be read by programs such as MOD2OBS, it can facilitate the use of MODFLOW 6 with PEST. However the loss or precision involved in double-to-single precision translation can result in some loss of integrity of finite-difference derivatives. Note also that the OLPROC program (which is documented in its own manual) was especially written to facilitate use of PEST with MODFLOW 6.

See Also

See also MANY2ONE, ARRDET, MOD2SMP, MOD2SMPDIFF and many others.

ELEV2CONC

Function of ELEV2CONC

ELEV2CONC was written to facilitate computation of initial concentrations for a multi-layer SEAWAT (or MT3D) model. It is assumed that the elevation of the freshwater/saltwater interface (or a surrogate for this interface calculated as, for example, a threshold concentration that is half way between that of freshwater and seawater) is available at a number of wells within a model area. It is also supposed that these wells provide information on the width of the transition zone between fresh and salt water. From these (together with other information from which the elevation of the interface may be surmised at other locations within the model domain or at its boundaries), a two-dimensional map of interface elevations may be made. From this information, a model-compatible real array of interface elevations can then be built. From this real array, together with interface width information, a three-dimension distribution of salt concentration can be established. This can then be supplied to a SEAWAT model as its concentration starting condition.

Using ELEV2CONC

Like all programs of the Groundwater Data Utility Suite, upon commencement of execution ELEV2CONC looks for a file named *settings.fig* in the directory from which its execution was invoked. This file informs ELEV2CONC whether (or not) a number-of-columns, number-of-rows header should be present in formatted real and integer arrays which it writes and reads. If this file is not present, ELEV2CONC ceases execution with an appropriate error message.

Also, if *files.fig* is present, ELEV2CONC will read the name of the grid specification file for the current model from this file. However it is not essential that this file be present.

After having checked for the presence of both of these files, ELEV2CONC requests information from the user through a series of prompts. As for all programs of the Groundwater Data Utilities suite, responding to any of these prompts by simply pressing “e” followed by <Enter> will force ELEV2CONC to backtrack to its previous prompt. This allows easy recovery from mistaken input.

ELEV2CONC’s first prompt is:-

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

ELEV2CONC needs to read this file so that it can obtain the horizontal dimensions of the model grid. Next it asks for the vertical dimension of the model grid.

```
Enter number of layers in model:
```

and then for the number of model layers for which initial concentrations are actually to be computed on the current ELEV2CONC run. These are bounded by an upper and lower model layer, the numbers for which must be supplied in response to the following prompts:-

```
Enter lower layer number for present analysis:
Enter upper layer number for present analysis:
```

(Note that a lower layer number overlies a higher layer number, as MODFLOW/MT3D grid layer numbers increase downwards from the surface.) ELEV2CONC must then ascertain which cells within the model domain are active, and which are inactive, for it needs to compute initial concentrations only within active cells. (It assigns inactive cells a concentration of zero.) As well as this, it needs to know which cells are constant concentration cells. As is the usual convention, these are identified through their negative values in integer activity arrays. So ELEV2CONC asks:-

```
Enter filename base for activity arrays:
```

Suppose that the response to the above prompt is the string "*activity*". ELEV2CONC then looks for arrays named *activityN.inf* where *N* ranges between N_u and N_l, these being the lower and upper model layers respectively for which initial concentrations must be computed. These arrays must be supplied in formatted (i.e. ASCII) form in these files. Furthermore, if the COLROW variable in *settings.fig* is set to "yes", these files must each contain a number-of-columns, number-of-rows header. (Note that these files can be written automatically on the basis of information contained within a MODFLOW or MT3D input file using the MOD2ARRAY utility.)

In order to translate freshwater/saltwater interface elevations to initial concentrations, ELEV2CONC must know the elevations of pertinent model layers. So it asks:-

```
Enter filename base of layer bottom elevation arrays:
```

Suppose that the response to the above prompt is the string "*bottom*". ELEV2CONC then looks for arrays named *bottomN.ref* where *N* ranges between N_u-1 and N_l, (The bottom of layer N_u-1 is assumed to be the top of layer N_u. If N_u is 1 then the elevation of the top of the model should be supplied in file *bottom0.ref* if the "*bottom*" string is supplied in response to the above prompt as discussed above.) Like activity arrays, these arrays can be extracted from a MODFLOW or MT3D input file using the MOD2ARRAY utility.

ELEV2CONC's next prompt is:-

```
Enter name of interface elevation array file:
```

Supply the name of a file containing the required real array. This array will normally have been produced through spatial interpolation of interface elevation measurements (see below); alternatively it may have been produced using the CONC2ELEV utility.

The freshwater/saltwater interface is by no means a sharp interface. On traversal of any vertical line which intersects this interface, the concentration rises gradually from that of fresh water to that of salt water as the interface is crossed. This vertical "smudging" of the interface must be reflected in initial concentrations supplied to the model. Three types of "smudging" are allowed by ELEV2CONC, namely linear, sigmoidal and exponential. The user chooses between these through his/her response to the following prompt.

```
Enter nature of concentration variation across interface:-
     if linear      - enter 1
     if sigmoidal   - enter 2
     if exponential - enter 3
```

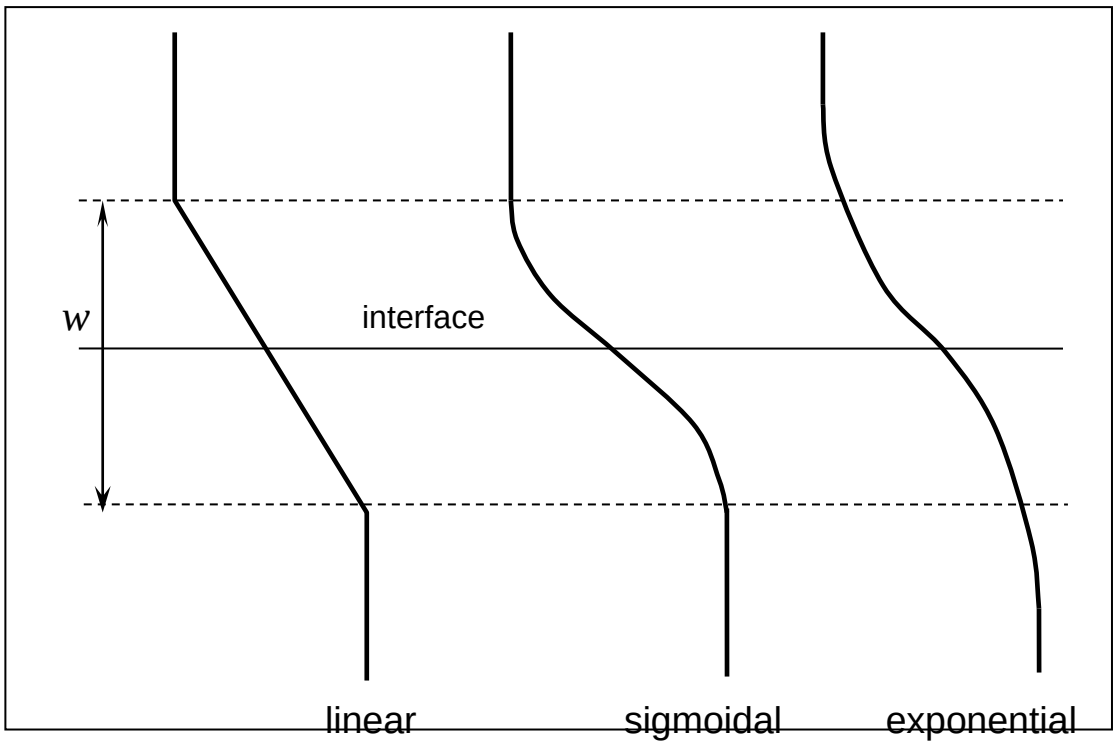
```
Enter your choice:
```

The actual width of the interface can be supplied as a single number (which is then applied throughout the entire model domain), or on a cell-by-cell basis. ELEV2CONC prompts:-

```
Enter interface width value or formatted real array file:
```

In response to this prompt supply a real number greater than zero, or the name of a file. If ELEV2CONC cannot interpret your response as a number it will presume that a filename has been supplied, and that this file contains a real array in formatted (i.e. ASCII) form. If the COLROW variable in *settings.fig* is set to "yes", then a number-of-columns, number-of-rows header is expected in this file.

The figure below illustrates the different types of concentration profile computed by ELEV2CONC.



Types of concentration profile across freshwater/saltwater interface.

If the "linear" option is selected, the concentration varies linearly over the distance between an elevation of *w/2* above the user-supplied interface elevation and *w/2* below the user-supplied interface elevation. Outside of these limits the concentration is that of freshwater (above) and sea water (below) respectively. Similarly, for the "sigmoidal" option, variations are limited to within a distance of *w*/2 of the interface, with concentration changes within this zone simulated as a quarter cycle of a sine wave. If the exponential option is chosen, concentrations approach those of fresh water and sea water asymptotically with ninety percent of the total change in concentration incurred over the interface width *w*. In all of these cases, the

concentration at the interface itself is half way between that of fresh water and sea water.

ELEV2CONC next asks for the concentrations of fresh water and sea water respectively. As for the interface width, either a single value can be supplied, or the name of a file which holds an array of cell-by-cell values can be supplied. As for the interface width, this must be an ASCII file, with a number of columns, number-of-rows header if the COLROW setting in *settings.fig* is set to "yes". The prompts are:-

```
Enter fresh water concentration value or formatted real array file:
Enter sea water concentration value or formatted real array file:
```

ELEV2CONC next asks:-

```
Enter filename base for formatted output files:
```

Suppose that the response to this prompt is "*conc*". Then ELEV2CONC writes a series of formatted real array files named *concN.ref* where *N* varies between N_u and N_l. Each such file carries concentrations in the specified (by *N*) layer. However as well as recording the outcomes of its concentration calculations in this format, ELEV2CONC also records these in unformatted (i.e. binary) form as an MT3D "UCN" file. It asks for the name of the file to which concentrations must be thus recorded using the following prompt:-

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

This file can be easily imported into a MODFLOW/MT3D graphical user interface for display, and for incorporation into a SEAWAT input dataset. (ELEV2CONC assigns a notional transport step, stress period, time step and simulation time of 1 to the concentration arrays recorded in this file.) However if initial concentration calculations have been restricted to a subset of model layers, ELEV2CONC cannot write this file unless it acquires knowledge of initial concentrations in other layers. So, if this is the case, it asks:-

```
Enter existing conc array filename base of unanalysed layers:
```

Suppose that a filename base of "*init*" is supplied in response to this prompt. Then ELEV2CONC will attempt to read formatted real array files *init1.ref*, *init2.ref*,…*initN.ref* etc where *N* is any integer value between 1 and NLAY (the number of layers in the model) EXCEPT for numbers between and including N_u and N_l. (This sequence of files may have been produced on previous ELEV2CONC runs.)

Once it has received all of this information, ELEV2CONC computes initial concentrations in all active cells in all nominated layers of the finite-difference grid. It then writes formatted concentration output files for all such layers, and its single binary output file in MT3D UCN format. Once it has finished these tasks it ceases execution.

The following item of ELEV2CONC functionality should be carefully noted. When assigning concentrations to cells within the model domain, all fixed concentration cells (identified through negative numbers in activity arrays) are assigned a concentration equal to that of sea water, this being based on the assumption that fixed concentration cells are always part of the seaward model boundary. If this is not the case, concentrations assigned to these cells by ELEV2CONC must be modified either by other utility software, or manually after importation of initial concentrations into a

graphical user interface. Alternatively, remove negative values from activity arrays before supplying them to CONC2ELEV.

Uses of ELEV2CONC

As stated above, the primary use of ELEV2CONC is computation of initial concentrations for a SEAWAT model. Proper computation of initial concentrations is important to this type of modelling, for then SEAWAT does not need to spend the first part of its run adjusting these concentrations from possibly infeasible initial values. Alternatively, if initial concentrations are computed on the basis of a steady state flow field, the time required for convergence to a corresponding concentration field can be reduced considerably if initial concentrations for this early part of the model run are close to those actually calculated on the basis of the flow field.

The freshwater/saltwater elevation array can be computed in a variety of ways based on interface elevations observed in wells, and on elevations of the coastal boundary condition where the freshwater/saltwater interface outcrops. Spatial interpolation can be carried out using, for example, the SURFER package; conversion to MODFLOW/MT3D real array format can then be implemented using the SRF2REAL utility. Alternatively, spatial interpolation could be undertaken using kriging as implemented by the PPK2FAC and FAC2REAL utilities described herein.

See Also

See also MOD2ARRAY, CONC2ELEV, SRF2REAL, PPK2FAC and FAC2REAL.

ELEV2CONC1

Function of ELEV2CONC1

The function of ELEV2CONC1 is almost identical to that of ELEV2CONC. However it computes (and records) a further set of output files, these being "zero flow head" arrays for a sequence of layers comprising a hydrostratigraphic unit of interest.

The "zero flow head" assigned to a cell is the head, computed for water that actually occupies that cell, that is equivalent to a salt water head of zero. Thus if:-

1. the lower boundary of a model domain is a coastal fixed head boundary at which the saltwater head is zero, and

2. there is actually no flow towards this boundary,

this would be the head in every active cell of the model.

ELEV2CONC1 was written for use in conjunction with SEAWAT. If ELEV2CONC1 zero flow head files are to serve as SEAWAT initial head input files (which they are designed to do), then cell-by-cell heads must be in accord with cell-by-cell salt concentrations; that is, head must be expressed in terms of the water that actually occupies the cell, irrespective of its concentration.

The relationship between head h and freshwater head h_f is defined by the equations (taken from the SEAWAT manual):-

$$h_f = \frac{\rho}{\rho_f} h - \frac{\rho - \rho_f}{\rho_f} Z$$

$$h = \frac{\rho_f}{\rho} h_f + \frac{\rho - \rho_f}{\rho} Z$$

where Z is the elevation of the measurement point (which are cell centres in the current context), ρ is the density of water at the measurement point, and ρ_f is the density of fresh water. The density of water is related to its concentration C by:-

$$\rho = \rho_f + k(C - C_f)$$

where C_f is the concentration of salt in fresh water and k is the slope of the density/concentration relationship (which is equal to 0.7143 if mass is measured in kg and volume is measured in m^3). If C_f is zero, this equation becomes:-

$$\rho = \rho_f + kC$$

With a little mathematical manipulation it is easy to show that the head in a cell whose salt concentration is C and whose salt water (defined as water with a concentration of C_s) head is zero is given by:-

$$h_{zf} = \frac{k(C - C_s)Z}{\rho_f + kC}$$

Using ELEV2CONC1

Use of ELEV2CONC1 is similar to that of ELEV2CONC. However ELEV2CONC1 asks four extra questions of the user, and provides an additional set of outputs. Defaults are provided for three of these four questions. These questions are:-

```
Enter density of water of zero concentration (<Enter> if 1000):
Enter concentration of water at zero head (<Enter> if 35):
Enter delta-density/delta-concentration slope (<Enter> if 0.7143):
```

The "concentration of water at zero head" is C_s in the above equations. Any cell within the model domain whose water concentration is C_s, is assigned a head of zero by ELEV2CONC1; for any cell with a lower concentration, the head is slightly higher than this (because of the lower density of the water).

The other ELEV2CONC1 prompt that is absent from ELEV2CONC is:-

```
Enter filename base for formatted "zero flow head" output files:
```

ELEV2CONC1 writes a series of formatted real array files, one for each of the layers on which ELEV2CONC1 calculations are focussed (these being specified by the user). Each of these files has the above user-supplied basename. A suffix equal to the layer number is appended to this name, followed by the extension ".ref" in accordance with the convention employed by programs of the Groundwater Data Utility suite for storage of formatted real arrays.

Uses of ELEV2CONC1

Uses of ELEV2CONC1 are similar to those of ELEV2CONC, for ELEV2CONC1 can do everything that ELEV2CONC can do, and more. The "more" is the production of "zero flow head" arrays.

ELEV2CONC1 can be employed for generation of initial concentration arrays for a SEAWAT model. If this is done, however, the problem remains of generation of initial heads arrays. If initial concentrations are assumed to be steady state (or approximately steady state), SEAWAT itself can be used to compute initial heads from initial concentrations if the initial stress period of the (coupled or uncoupled) SEAWAT model is steady state from the MODFLOW standpoint. Computation of initial heads by SEAWAT in this fashion requires, however, that heads supplied to it are correct in fixed head cells. Where the only fixed head cells in a model are along the coast, and where the saltwater heads in these cells are zero, the set of heads provided by ELEV2CONC1 are suitable starting heads from which SEAWAT can compute a steady state head field corresponding to the concentration field which ELEV2CONC1 produces.

See Also

See also CONC2ELEV and ELEV2CONCl.

FAC2FEFL

Function of FAC2FEFL

FAC2FEFL complements the PPK2FAC_FEFL utility in that it undertakes spatial interpolation from pilot points to elements of a FEFLOW mesh on the basis of kriging factors computed by PPK2FAC_FEFL. As such, it is often run as part of a "composite model" calibrated by PEST in which pilot point values for one or more material properties are estimated.

Using FAC2FEFL

Prompts and User Responses

Use of FAC2FEFL is similar to that of other programs (for example FAC2REAL, FAC2RSM etc) which undertake spatial interpolation to a model grid or mesh based on pre-calculated kriging factors. Hence its use will not be described in detail herein. The description below focuses on differences between FAC2FEFL and other programs of the same type. See documentation for these other programs for further details.

Typical FAC2FEFL screen output and user responses (the latter in bold italicised print) are provided below.

```
Program FAC2FEFL carries out spatial interpolation for a FEFLOW model based
  on interpolation factors calculated by PPK2FAC_FEFL and pilot point
  values contained in a pilot points file.

Enter name of interpolation factor file: factors.dat
Is this a formatted or unformatted file? [f/u]: f

Enter name of pilot points file [pp.dat]: pp.dat
 - data for 27 pilot points read from pilot points file pp.dat

Enter lower interpolation limit: 1e-10
Enter upper interpolation limit: 1e10

Enter name of existing FEM file: model.fem
Enter code for element list identification: ###
Enter name for new FEM file: temp.fem

- interpolating from pilot points to finite element mesh...
- reading existing FEM file and writing new one...

- file model.fem read ok.
- file temp.fem written ok.
```

The interpolation factor file read by FAC2FEFL must have been written by the PPK2FAC_FEFL utility. Furthermore, it should have been produced on the basis of the same set of pilot points from which interpolation is being implemented by FAC2FEFL. (FAC2FEFL will generate an error message to this effect if this is not the case.)

FAC2FEFL assigns hydraulic property values to mesh elements. It then writes these values to a FEFLOW "*fem*" file. In doing this, it replaces hydraulic property values already recorded in such a file with new ones obtained through the spatial interpolation process. Thus it reads an existing "*fem*" file, and writes a new one. Details of this process are now discussed.

The "fem" File

In specifying values assigned to mesh elements for a certain hydraulic property, the "*fem*" file first provides a header in which the hydraulic property type is cited. Then follows a "node list" in which hydraulic property values assigned to individual elements are specified. Two examples are provided below. The first illustrates non-uniform material properties, whereas the second illustrates uniform values for each cited property type. Notice that in both cases a real number (representing a hydraulic property value) is followed by the list of elements to which the property value pertains. Note also that a dash between element numbers indicates that all elements between (and including) the elements on either side of the dash are assigned the cited property value.

```
MAT_I_FLOW
101 0.000000e+000 "Conductivity in x-direction for 3D"
  8.640000e-010            3    40    47    54    57    60    61   151   178
                438   453   471   473   475   476   478   480   487   489   511   512
                691   793   794   845   850   895   936   949   951-  953   955
                961-  966   997  1016  1017  1029  1033  1140  1208  1211  1215  1219
               1222  1225  1573  1617  1619  1620  1627- 1630  1633- 1635  1637  1638
               1780  1806  1881  1966  2152  2203  2211  2218- 2224  2227  2239  2263
               2275  2276  2280  2281  2288  2290  2292  2307  2341  2378  2389  2393
               2403  2415  2427  2442  2510  2591  2607  2640  2658  2665  2700  2705
etc
```

Material properties provided through a node list.

```
MAT_I_FLOW
101 0.000000e+000 "Conductivity in x-direction for 3D"
  1.345678e-006           1-92829
103 0.000000e+000 "Conductivity in y-direction for 3D"
  4.7845673-005           1-92829
105 0.000000e+000 "Conductivity in z-direction for 3D"
  4.7845673-005           1-92829
etc
```

Material properties provided through a node list.

Before running FAC2FEFL, the user must edit the "*fem*" file for the current model in order to provide a means of informing FAC2FEFL of the hydraulic property type(s) for which an existing node list must be replaced by a new one. This is achieved through inserting an "element list identification" code on the first line of the respective node list. This is a string of five characters or less which identifies the list as subject to replacement. See the examples below (corresponding to those above) for which, in each case, the element list identifier is "###". (Note that this identifier is not transferred to the new "*fem*" file which FAC2FEFL writes.)

```
MAT_I_FLOW
101 0.000000e+000 "Conductivity in x-direction for 3D"
  8.640000e-010           3    40    47    54    57    60    61   151   178 ###
                 438   453   471   473   475   476   478   480   487   489   511   512
                 691   793   794   845   850   895   936   949   951-  953   955
                 961-  966   997  1016  1017  1029  1033  1140  1208  1211  1215  1219
                1222  1225  1573  1617  1619  1620  1627- 1630  1633- 1635  1637  1638
                1780  1806  1881  1966  2152  2203  2211  2218- 2224  2227  2239  2263
                2275  2276  2280  2281  2288  2290  2292  2307  2341  2378  2389  2393
                2403  2415  2427  2442  2510  2591  2607  2640  2658  2665  2700  2705
etc
```

Material properties provided through a node list, and an element list identification code.

```
MAT_I_FLOW
101 0.000000e+000 "Conductivity in x-direction for 3D"
  1.345678e-006          1-92829                        ###
103 0.000000e+000 "Conductivity in y-direction for 3D"
  4.7845673-005           1-92829                        ###
105 0.000000e+000 "Conductivity in z-direction for 3D"
  4.7845673-005           1-92829
etc
```

Material properties provided through a node list, and element list identification codes.

The following should be noted.

1. The element list identifier must be placed on the first line of the node list; it must not be placed on the hydraulic property header above this line, or on the second (or subsequent) lines of the node list.

2. The sequence of up to five characters comprising an element list identifier must occur nowhere else within a "*fem*" file, so that it can provide a unique means of identifying replaceable lists.

3. Element list identifiers are easily added to a FEFLOW "*fem*" file using a text editor. However the text editor must be capable of reading large files. It must also perform no gratuitous alterations to any other part of the file (e.g. replacement of tabs with spaces).

4. More than one incidence of the same element list identifier can be placed in a single "*fem*" file. Node list replacement on the basis of pilot-point-interpolated values will then take place for all such identified node lists.

In replacing the identified node list with a new one, FAC2FEFL observes the following protocols. These should be carefully noted.

1. If no kriging factors have been computed for a particular mesh element, the value assigned to that element in the existing node list is unaltered as the list is transferred to the new "*fem*" input file.

2. Whether or not interpolation factors have been ascribed to a particular mesh element, if that element is not cited within the existing node list, it will not be cited in the new node list.

3. No aspect of the existing "*fem*" file, other than node lists identified as above, is altered in building the new "*fem*" file from the existing one.

Running FEFLOW

If FEFLOW is to be run repeatedly by PEST as part of a (possibly lengthy) calibration process, its execution must be activated using a system command. FEFLOW can, in fact, be run from outside its interface using a command such as the following:-

```
feflow -run -hide -work c:\feflow -steps tstep1.pow -dar temp.dar new.fem
```

Run FEFLOW using the command:-

```
feflow –help
```

for usage details. Note that for both of the above commands, either the "feflow" string should be prefixed by the name of the directory containing the FEFLOW executable, or this directory should be including in the PATH environment variable.

Uses of FAC2FEFL

FAC2FEFL, together with its sister program PPK2FAC_FEFL, comprises the mechanism through which pilot points can be employed for parameterisation of a FEFLOW model. This, in turn, can promulgate the use of more parameters in the calibration process than would otherwise be possible. As is described in PEST and ancillary documentation, the use of many parameters, rather than just a few, allows mathematical regularisation to be introduced to the inversion process. This in turn can lead to extraction of maximum information content from a calibration dataset (thereby leading to model predictions of greater likelihood), and the ability to quantify the potential error associated with such predictions.

The DAR2SMP utility provides a means through which FEFLOW outputs can be rapidly processed for inclusion in a PEST-based calibration process.

See Also

See also PPK2FAC_FEFL, FAC2REAL, FAC2FEM, FAC2RSM and DAR2SMP.

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.

FAC2G

Function of FAC2G

FAC2G carries out spatial interpolation based on kriging factors computed by the PPK2FACG utility. "G" stands for "general", as no particular model is assumed. As explained in documentation of the PPK2FACG utility, the format for the file which lists the points to which interpolation must take place (these normally being the nodes of a numerical grid or mesh) is comprised simply of three columns of data. The format of the file written by FAC2G is even simpler than this, being comprised of a single column of numbers, these being the outcomes of interpolation to the specified points. Points are arranged in the same order as that provided to PPK2FACG when it calculated kriging factors.

Using FAC2G

FAC2G prompts, and typical responses to these prompts, are as follows.

```
Enter name of interpolation factor file: factors.dat
Is this a formatted or unformatted file? [f/u]: f

Enter name of pilot points file [vk5.pts]: vk5.pts
 - data for 193 pilot points read from pilot points file vk5.pts

Enter lower interpolation limit: -1e20
Enter upper interpolation limit: 1e20

Enter name for output file: temp1.dat
Enter value for elements to which no interpolation takes place: 1e35
```

Reasons why interpolation to a specific point may not take place include the following.

- The point lies outside the interpolation radius of all pilot points.
- The point does not lie within a zone to which any pilot points belong.
- The user specified when running PPK2FACG that no interpolation takes place to points within the zone to which that point belongs.

See Also

See also PPK2FACG 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.

See Also

See FIELDGEN, PPK2FAC and PPKREG.

FAC2REAL3D

Function of FAC2REAL3D

FAC2REAL3D undertakes three-dimensional kriging, interpolating parameter values listed in a three-dimensional pilot points file to cell-centres of a MODFLOW finite-difference grid using kriging factors computed by the PPK2FAC3D utility.

Using FAC2REAL3D

Use of FAC2REAL3D is similar to that of FAC2REAL.

Like other members of the Groundwater Data Utility suite, the user can backtrack to previous prompts by responding to any prompt with "E" or "e" followed by <Enter>.

Upon commencement of execution FAC2REAL3D looks for a settings file named *settings.fig* in the directory from which its execution was initiated. It obtains a COLROW setting from this file. If *settings.fig* is not present, a COLROW setting of "no" is presumed.

FAC2REAL3D's first prompts are:

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

Respond as appropriate. Next FAC2REAL3D asks:

```
Enter name of 3D pilot points file [file.dat]:
```

where *file.dat* is the name of the three-dimensional kriging file provided to PPK2FAC3D on the basis of which it computed kriging factors. Press <Enter> to accept this filename, or supply the name of another pilot points file. However if the latter course is followed, the file must cite the same pilot points (including the same names) in the same order as that in the file provided provided to PPK2FAC3D when it computed the kriging factors, for fast spatial interpolation depends on this. If this is not the case PPK2FAC3D will terminate execution with an appropriate error message.

FAC2REAL3D next asks for interpolation limits:

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

Provide appropriate numbers in response to these prompts. If an interpolated value at any cell within the three-dimensional model domain is less than or greater than (respectively) these limits, this value will be clipped in order to respect the limits.

FAC2REAL3D's next prompt is:

```
Write outputs to single 3D table or multiple 2D real array files? [s/m]:
```

If the user's response is "m" FAC2REAL3D then asks:

```
Enter filename base for output real array files:
```

Suppose that the response to this prompt is, for example, *kx*. Then FAC2REAL3D will write a series of files named *kx1.ref, kx2.ref....kxN.ref* where *N* is the number of layers in the model domain. Each of these files holds a two-dimensional real array; collectively they span the model grid. On the other hand, if the user's response to the above prompt is "s", FAC2REAL3D asks:

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

On the first line of this output file FAC2REAL3D records three integers, these being the number of columns, rows and layers respectively comprising the MODFLOW grid. It then lists the value interpolated to every cell in the grid, with a single entry on each line. These entries are listed with column numbers cycling fastest, then row numbers and then layer numbers.

Finally FAC2REAL3D prompts:

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

Cells which belong to zones for which no spatial interpolation has been requested, or are further removed from the nearest pilot point than the maximum search radius specified when PPK2FAC3D was run will not have interpolated values assigned to them. It is normally best to respond to the above prompt with a high value such as 1.0e35 so that these cells are easily recognized.

Uses of FAC2REAL3D

FAC2REAL3D is employed in conjunction with PPK2FAC3D for undertaking three-dimensional interpolation.

See Also

See also FAC2REAL and PPK2FAC3D.

FAC2RSM

Function of FAC2RSM

FAC2RSM builds an input data file for the RSM model (i.e. "Regional Simulation Model") developed by the South Florida Water Management District by undertaking spatial interpolation from a set of pilot points to the model mesh. In this respect it is the RSM-equivalent of FAC2REAL (which undertakes the same task in the MODFLOW/MT3D context). Its use is predicated on the assumption that kriging factors have been previously computed by PPK2FACR.

Using FAC2RSM

Use of FAC2RSM is similar to that of FAC2REAL. Its use will therefore be described only briefly; the user is referred to FAC2REAL for further documentation.

FAC2RSM commences execution with the prompts:-

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

FAC2RSM obtains PPK2FACR-generated kriging factors from this file. Next it prompts:-

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

Sometimes the above prompt will be accompanied by a default filename, this name having been obtained from the PPK2FACR-generated kriging factors file. If so, it can be accepted by simply pressing the <Enter> key. FAC2RSM obtains values which it must interpolate to the mesh from the pilot points file.

FAC2RSM next prompts:-

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

No interpolated value, in any part of the model mesh, will be allowed to undercut the value supplied in response to the first of the above prompts, or exceed the value supplied in response to the second of the above prompts. (Note that these limits are provided only as a check on the sometimes spurious values that can result from kriging – a phenomenon that is more likely to occur when kriging is based on some variograms - e.g. the Gaussian variogram – than others. These values should not be used to limit parameter ranges during the parameter estimation process. This should be done using upper and lower parameter bounds in the PEST control file.)

FAC2RSM next asks for the name of the file that it should write. Its prompt is:-

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

Then it asks:-

```
Enter NAME of data type:
```

This text string provided here will be inserted after the NAME keyword on the fifth line of the RSM input data file. If the name is comprised of two words, then these words should be encased in quotes.

FAC2RSM's final prompt is:-

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

These will be mesh elements which were assigned to a zone for which no kriging factors were computed when PPK2FACR was run prior to FAC2RSM.

Once it has received all of the above information FAC2RSM writes the nominated RSM input file, computing mesh element values through interpolation from pilot point values.

Uses of FAC2RSM

FAC2RSM comprises part of the means through which pilot point parameterisation can be employed for RSM. These parameters can be estimated by PEST through regularised inversion. At least one instance of FAC2RSM will comprise part of the model run by PEST during the inversion process.

See Also

See also PPK2FACR, RSM2SRF and RDAT2TAB.

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 20 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 20 characters in length, FEM2SMP shortens them (from the left) to 20 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 20 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 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 Part A of this manual.

FIELDGEN first prompts for the name of a grid specification file (see 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 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 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 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. Note also that, irrespective of whether or not a log variogram is employed for field generation, the value supplied here must pertain to native rather than log-transformed values. (It is log-transformed internally.)

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 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 FIELDGEN_SVA, FIELDGEN3D_SVA, FAC2REAL and PPK2FAC.

Reference

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

FIELDGEN_SVA

Function of FIELDGEN_SVA

Like FIELDGEN, FIELDGEN_SVA generates stochastic hydraulic property fields for a single layer of a structured MODFLOW model. However similarities with FIELDGEN end with this.

Unlike FIELDGEN, FIELDGEN_SVA does not allow conditioning of stochastic fields using point measurements of system properties. However conditioning can be achieved in other ways; see below. Nor does FIELDGEN_SVA allow a user to partition a model domain into zones that define areas of differing hydraulic property field stochasticity. However FIELDGEN_SVA offers more flexible options for expressing spatial variability of stochasticity.

FIELDGEN_SVA abandons the notion of stationarity. The sill, range, anisotropy and anisotropy bearing of the variogram on which stochastic field generation is based can be different in every cell of a model domain. However use of the term "variogram" is not quite correct, for while the stochastic field which FIELDGEN_SVA generates can be conceptualized as being characterised by a spatially-varying variogram, the actual variogram is unknown.

Before describing the use of FIELDGEN_SVA, some theory will be presented. For more details see the following papers, and references cited therein.

Higdon, D, Swall, J. and Kern J., 1999. Non-stationary spatial modeling, in: *J.M. Bernardo et al. (Eds.), Bayesian Statistics 6, Oxford University Press, Oxford*, pp. 761–768.

Fuentes, M., 2002. Spectral methods for nonstationary spatial processes. *Biometrika*, 89: 197-210.

Oliver, D., 2022. Hybrid iterative ensemble smoother for history-matching of hierarchical models. *Math. Geosci*. 54: 1289-1313.

Some Theory

General

FIELDGEN_SVA employs the so-called "non-centred" method of stochastic field generation. A benefit of this method is that it can be used to generate stochastic fields based on both stationary and non-stationary variograms. A disadvantage of the method is that the actual variogram that is associated with the generated stochastic field is unknown, except in special circumstances. This matters if a variogram that is derived from field measurements of hydraulic properties must be respected. However it generally does not matter for stochastic fields that are used by a groundwater model, for the spatial correlation length of these fields is guessed on most occasions. If a

modeller does not like the patterns of spatial variability that FIELDGEN_SVA generates, then he/she is at liberty to regenerate these fields using different FIELDGEN_SVA settings until the patterns reflect those which he/she expects to prevail in the real world. Nevertheless, as we shall see, approximate relationships between FIELDGEN_SVA control variables and correlation lengths exhibited by the stochastic patterns that it generates are reasonably easy to establish.

Spatial Averaging

Suppose that FIELDGEN_SVA is asked to build a stochastic field for a grid whose column and row dimensions are *ncol* and *nrow*. FIELDGEN_SVA begins by sampling a standard, univariate normal probability distribution *ncol* × *nrow* times. This distribution has a mean of zero and a standard deviation of 1.0. FIELDGEN_SVA assigns one such random sample to each model cell. Next, for every grid cell to which it must assign a random hydraulic property value, it passes a moving average over the independent standard normal deviates that occupy the grid; this is equivalent to a two-dimensional convolution operation.

It can be shown that if this moving average filter is proportional to a Gaussian function - i.e. if it is proportional to $exp[-(h/a)^2]$ then the resulting stochastic hydraulic property field is equivalent to that which would be generated using a Gaussian variogram with an "a" value of $\sqrt{2}$ times that which is used for the spatial averaging function. An appropriate constant is applied in order to achieve the desired variogram sill.

Convolution using other spatial averaging functions yields stochastic hydraulic property fields that pertain to other variograms. However, in these cases, the variogram may not have an analytical expression. Spatial correlation that is exhibited by a stochastic field which is generated in this way extends to a length which is commensurate with the length over which the convolution function (i.e. the moving average function) is nonzero. In actual fact, the range of the implied variogram is somewhat greater than the range of the moving average function. This is because two points in space exhibit spatial correlation if moving average functions which are centred on each of them include the same grid point, Hence the same sampled random normal variate is included in the respective spatial averages that generate hydraulic property values at both of these points.

Spatially Varying Stochasticity

The moving average method is easily extended to nonstationary contexts. The moving average function can therefore vary in space. Hence the function which is used to evaluate the hydraulic property to one point in a grid can be different from that which is used to evaluate the hydraulic property at other points in the grid. The variables which govern the grid-point-specific moving average function can be spatially interpolated to all model grid cells prior to implementing the spatial averaging process.

The Functions

FIELDGEN_SVA employs the following spatial averaging functions. Note that while some of these functions have the same names as variograms, these names do not describe the variograms that pertain to the fields that they generate (except for the Gaussian function) for reasons that are described above. Note that each of these functions is equal to 1.0 when the abscissa *h* is zero. (*h* is the distance between the target point for which the random hydraulic property value is calculated and the field point; averaging takes place over field points).

FIELDGEN_SVA allows the user to assign a variance to each grid point. Coefficients assigned to the spatial averaging function ensure that this variance is respected. Note also, that, as is discussed below, FIELDGEN_SVA supports variogram anisotropy. Where anisotropy prevails, the value of *h* in the following equations is direction-dependent.

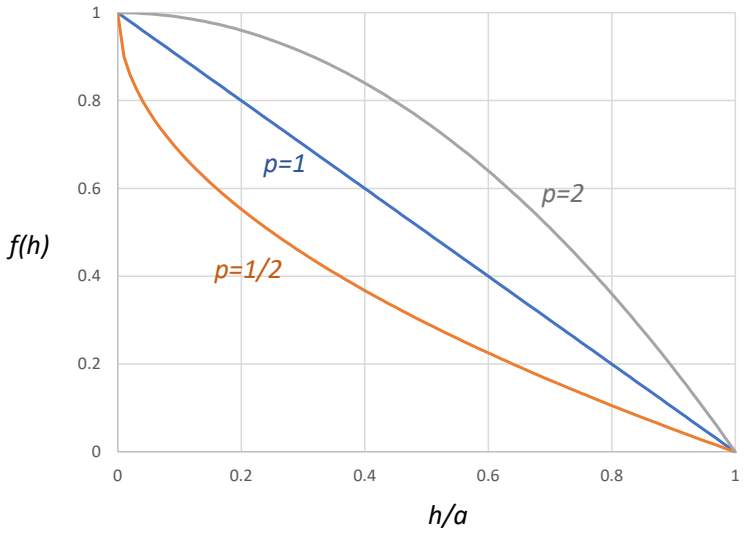
Spherical: $f(h) = 1 - \frac{3}{2}\frac{h}{a} - \frac{1}{2}\frac{h^3}{a^3}$

Exponential: $f(h) = exp\left(-\frac{h}{a}\right)$

Gaussian: $f(h) = exp\left(-\frac{h^2}{a^2}\right)$

Power: $f(h) = 1 - \left(\frac{h}{a}\right)^p$

The following picture depicts the power function for different values of the power *p*.



The power averaging function for different values of power.

Which is the best averaging function to use? The user must decide for him/herself based on the patterns that *FIELDGEN_SVA* generates. Note, however, that calculation of convolution integrals for the exponential averaging function (and to some extent the Gaussian averaging function) may be slower than for the other functions. For the spherical and power functions, averaging ceases at a distance of *h/a* from each target point. For the exponential and Gaussian averaging functions it can extend further than this. For large, dense model grids this can increase the time required for populating the grid with a random hydraulic property field.

Using FIELDGEN_SVA

FIELDGEN_SVA asks a lot of questions. In common with all other members of the Groundwater Utility Suite, if you respond to any prompt with "e", you are taken back to the previous prompt.

In common with many other programs of the Groundwater Utility Suite, FIELDGEN_SVA expects to find a settings file named *settings.fig* in the folder from which it is run. This file specifies whether an NCOL NROW header precedes arrays that are read from text (but not binary) files.

FIELDGEN_SVA begins execution with the following prompt:

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

It is assumed that the grid is structured. The grid specification file is therefore easy to build; see part A of this manual for its specifications. (Many programs of the Groundwater Utility suite begin execution with this same prompt.)

FIELDGEN_SVA next asks for an activity array. It will only calculate random hydraulic properties for grid cells whose IBOUND value is non-zero. The prompt is:

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

FIELDGEN_SVA next asks for an array that provides the mean hydraulic property value in every cell of the grid. Random values are calculated as departures from this mean. The mean can be provided as a single number or as a file; if FIELDGEN_SVA cannot read the response to this prompt as a number, it assumes that the response is the name of a file. The array that is contained in this file must be compatible with the grid that is defined in the grid specification file. If the user-supplied filename has an extension of ".*reu*", then the file is presumed to be binary; all other extensions are taken to imply a text file. The prompt is:

```
Enter uniform mean hydraulic property value or array filename:
```

The next prompt is:

```
Express stochasticity in natural or log domain? [n/l]:
```

If the response to this prompt is "l" then variables which govern stochastic field generation are assumed to pertain to the log (to base 10) of the hydraulic property for

which random field realisations are required. After a stochastic field is generated, every element of it is raised to the power of 10. It then multiplies respective elements of the mean hydraulic property field. Alternatively, if the response to the above prompt is "n", random parameter fields that are generated by FIELDGEN_SVA are added to the user-supplied mean hydraulic property field.

FIELDGEN_SVA now asks a number of questions that pertain to the spatial averaging function. The answer to each of these questions can be provided as a single number, or as the name of a file that contains a grid-compatible array. The prompts are:

```
Enter uniform variance/sill or array filename:
Enter uniform "a" or array filename:
Enter uniform anisotropy ratio or array filename:
Enter uniform anisotropy bearing or array filename:
```

Note the following:

- It is apparent from the first of the above prompts that the variance of the hydraulic property field generated by FIELDGEN_SVA can vary on a cell-by-cell basis.

- The "a" value pertains to the above equations for the spatial averaging function. This determines the (spatially variable) spatial correlation length.

- Supply a value of 1.0 for no averaging function anisotropy; the random field that is generated by FIELDGEN_SVA will therefore be isotropic. Otherwise supply a value that is greater than 1.0.

- The anisotropy bearing (which is 0.0 degrees for north and 90.0 degrees for east) must lie between -360 degrees and 360 degrees. This pertains to the direction of largest spatial correlation length. That is, it is the bearing of the major axis of the anisotropy ellipse.

FIELDGEN_SVA next asks the user which of the above functions should be used for spatial averaging. The prompt is:

```
Is moving ave spherical, exponential, gaussian or power? [s/x/g/p(..)]:
```

If a power function is desired, then the response to the prompt should be something like: "`p(1.0)`". This indicates a power of 1.0. Alternatively a response of "`p(0.5)`" indicates a power of 0.5. The power must be greater than zero. A power of 1.0 defines a linear function.

FIELDGEN_SVA's next prompt is as follows:

```
Enter number of grid buffer cols/rows (<Enter> if none):
```

Spatial averaging comes to an end at the edges of a grid. Hence, for target cells which are at or near the edge of a grid, the averaging function process is truncated. In theory, this may distort near-border random field values. Hence the grid over which

independent normal variates are integrated can be extended beyond the model grid. Enter the number of cells for which the grid should be extended in response to the above prompt. (Experience to date suggests that this is not necessary.)

FIELDGEN_SVA's next three prompts are as follows:

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

Suppose that the response to the second of the above prompts is "test" and that the response to the third of the above prompt is "f". Then FIELDGEN will store the random arrays that it generates in files named *test1.ref, test2.ref….test100.ref*. Alternatively, if the response to the third prompt is "u" then binary files named *test1.reu, test2.reu…test100.reu* are generated. Note that for inactive cells the values that are recorded in all of these files are the same as those that are supplied by the user as "mean values" for those cells.

FIELDGEN_SVA's next prompts pertain to strategies that can increase the numerical efficiency of random field generation. First:

```
Summation factor cutoff threshold (<Enter> if 1.0E-5):
```

This question is only relevant where an exponential or Gaussian averaging function is employed. FIELDGEN_SVA uses the response to this question to limit the area over which spatial averaging takes place. Suppose that the averaging function is exponential, and that the response to the above prompt is 1E-5. Then FIELDGEN_SVA limits the integration area to the distance from each target cell at which exp(*h/a*) is greater than 1E-5. As stated above, for the spherical and power averaging functions, the integration distance is automatically limited to a *h/a* value of 1.0.

```
Array processing block size (<Enter> if 5):
```

Suppose that the response to this prompt is "5". Then FIELDGEN_SVA generates 5 random hydraulic property fields at a time. Spatial convolution is performed in parallel for these 5 fields. Suppose that 100 random parameter fields are requested. Then this spatial convolution process is repeated 20 times.

The cost of this efficiency measure lies in the memory that it requires. If the block size is 5, then 10 model-compatible arrays must be stored at once by FIELDGEN_SVA. Five of these arrays contain independent random normal variates while the other five are filled through spatial averaging of these variates. If the grid is not large, this is a small cost to pay for efficiency. The response to the above prompt may then be increased to the number of realisations for which stochastic field generation is required.

Finally, FIELDGEN_SVA prompts for a random seed:

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

Provide a positive integer, or press <Enter>.

FIELDGEN_SVA then does the work requested of it, recording its progress to the screen as it does so. It then terminates execution.

Conditioning

A disadvantage of random field generation through moving averaging is that random fields which are produced in this way are not readily conditioned by point measurements of hydraulic properties. (The opposite is the case for the sequential Gaussian field generation method employed by FIELDGEN; as is discussed in documentation of this program, FIELDGEN allows a user to supply a file that contains conditioning points.)

Conditioning can be effected by FIELDGEN_SVA in a more roundabout manner. Suppose that there are *N* points within the model grid at which hydraulic property values have been measured. Then the mean parameter field value that is supplied by the user should be equal to these observed values at the *N* measurement point locations. At the same time, the user-supplied variance should be a small value at these and surrounding points in order to express the conditioning effect of these points.

Providing Arrays

As is discussed above, FIELDGEN_SVA allows the user to optionally supply the following variables as arrays:

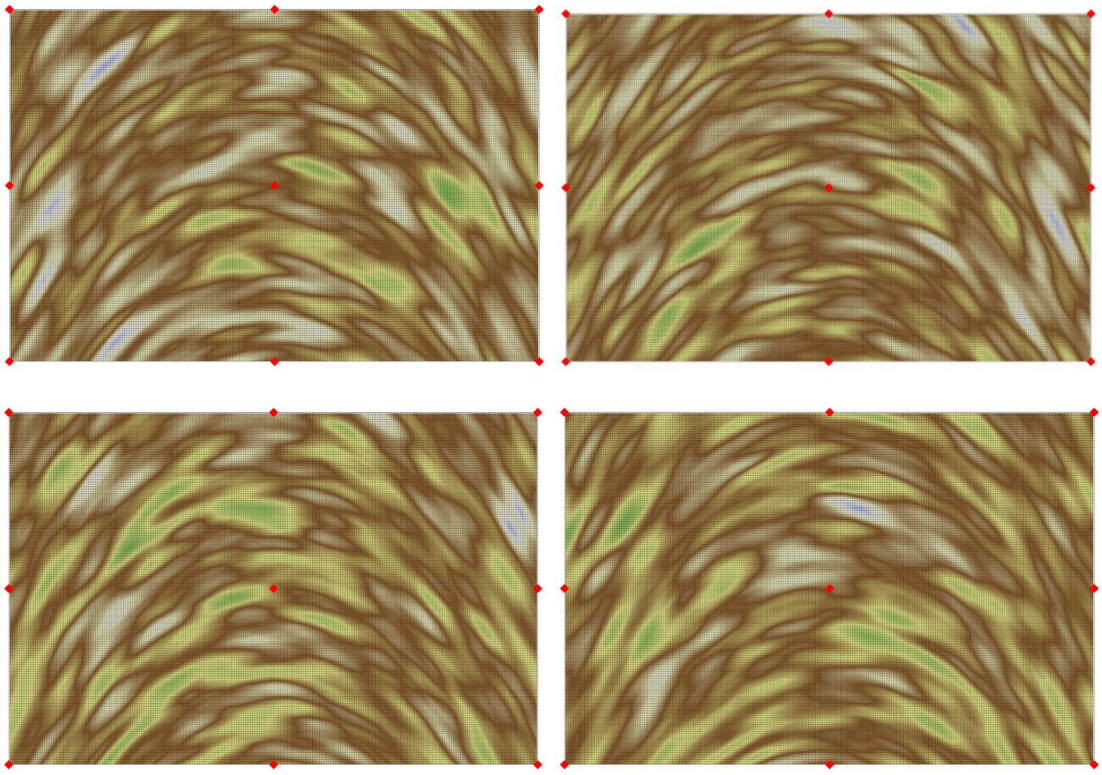
- Mean hydraulic property field;
- Variance of random hydraulic property fields:
- "a" value of the spatial averaging function;
- Anisotropy ratio;
- Anisotropy bearing.

Values for these variables can be readily interpolated from a set of points using the PLPROC parameter preprocessor. PLPROC allows the outcomes of this interpolation process to be recorded as MODFLOW-compatible real arrays which are easily read by FIELDGEN_SVA.

Example

The following random fields were generated using FIELDGEN_SVA. The grid is 200 rows by 300 columns. The spatial averaging function is of the power type; the power is 1.0 (i.e. linear). Variances, *a* values, anisotropy ratios and anisotropy bearings are supplied as arrays; the values which comprise the elements of these arrays were spatially interpolated (using PLPROC) from the red points. The anisotropy ratio

varies between 4.0 and 5.0. The anisotropy bearing varies between 60 degrees (at the left of the grid) to 150 degrees (at the right of the grid).



See Also

FIELDGEN, PLPROC.

FIELDGEN2D_SVA

Function of FIELDGEN2D_SVA

FIELDGEN2D_SVA is the two-dimensional equivalent of FIELDGEN3D_SVA. It uses spatial averaging to generated a two-dimensional stochastic hydraulic property fields for a structured or unstructured grid model. The properties of the variogram on which these fields are based can vary from cell to cell within the model domain.

Some Theory

See documentation of FIELDGEN_SVA for the theory on which stochastic field generation using spatial averaging is based.

File Types

FIELDGEN2D_SVA requires two tabular data files. These can be CSV files if you like; that is, the contents of these files can be space or comma-delimited.

2D Node Specification File

The first of these file types is referred to as a "2D node specification file". Its contents pertain to nodes of a model grid. FIELDGEN2D_SVA does not care whether the model grid is structured or unstructured.

The figure below shows the first few lines of such a node specification file displayed in Microsoft EXCEL.

node	x	y	area	active
1	466082.499	7490090.611	62500	0
2	466294.511	7489958.131	62500	0
3	466506.523	7489825.651	62500	0
4	466718.535	7489693.172	62500	0
5	466930.547	7489560.692	62500	0
6	467142.559	7489428.212	62500	0
7	467354.571	7489295.732	62500	0
8	467566.583	7489163.252	62500	0
9	467778.595	7489030.772	62500	0
10	467990.607	7488898.293	15625	0
11	468202.619	7488765.813	15625	0
12	468414.631	7488633.333	15625	0
13	468626.643	7488500.853	15625	0
14	468838.655	7488368.373	15625	0

First few lines of a 2D node specification file.

FIELDGEN2D_SVA insists that certain protocols be followed by this file. The first line of a 2D node specification file must contain the same headers as those that appear in the above picture. These define the contents of respective columns. The first column must contain model node numbers. FIELDGEN2D_SVA insists that these be numbered sequentially starting from 1. The *x* and *y* coordinates of nodes (normally cell centres) follow in ensuing columns. The next column contains cell areas. The final column specifies whether a cell is active or not. FIELDGEN2D_SVA does not calculate stochastic property field values for inactive cells.

Note that cell areas are used in the integration procedure through which the spatial averaging function is applied.

2D Averaging Function Specification File

The first part of a 2D averaging function specification file is pictured below. In many instances of its use, the contents of this file will have been spatially interpolated to the locations of model grid cell centres.

node	mean	sill	aa	anis	bearing
1	10	0.25	1000	1	0
2	10	0.25	1000	1	0
3	10	0.25	1000	1	0
4	10	0.25	1000	1	0
5	10	0.25	1000	1	0
6	10	0.25	1000	1	0
7	10	0.25	1000	1	0
8	10	0.300667	1035.102	1.534945	14.73745
9	10	0.277463	771.7609	1.273803	23.89539
10	10	0.268254	686.4736	1.177781	26.61751
11	10	0.290968	940.5336	1.422418	25.51952
12	10	0.279487	840.2938	1.295487	21.26855
13	10	0.259086	617.8432	1.086419	17.1166
14	10	0.257649	661.115	1.072481	13.42673
15	10	0.270535	790.1712	1.201165	10.28369
16	10	0.264596	797.177	1.140823	8.067899
17	10	0.258932	750.769	1.084921	23.74899
18	10	0.261892	854.681	1.113938	27.00724
19	10	0.260699	925.6274	1.102191	26.30784
20	10	0.251098	510.6087	1.01023	21.60985
21	10	0.254521	596.3752	1.042494	17.2113

The first part of a 2D averaging function specification file.

A 2D averaging function specification file must possess six columns. Their labels should be exactly as shown in the above figure. (However you can use “variance” instead of “sill” and “a” instead of “aa”.) The contents of these columns are obvious from their labels. As for the 2D node specification file, the first column must feature

grid nodes. These must be sequential and start at 1. All grid nodes must be cited in this column whether or not they are active.

Note that bearing is measured clockwise from north. Anisotropy ratio is generally greater than 1.0, but not does not have to be so. It is the ratio of continuity in the principal direction of anisotropy (that is, in the *bearing* direction) to that in directions that are perpendicular to this.

Note that a user may wish to supply data for only active nodes of a model grid in the 2D node and averaging function specification files. Hence some nodes of a model grid may be omitted from both of these files. FIELDGEN2D_SVA does not care about this, for it has no knowledge of the grid structure. However, if a user chooses to do this then he/she must still ensure that nodes are numbered sequentially starting at 1 in both of the files that are supplied to FIELDGEN2D_SVA, and that nodes in these two files correspond to each other.

Using FIELDGEN2D_SVA

FIELDGEN2D_SVA begins execution by prompting for the names of the above two files. It then reads these files, reporting any errors that it encounters to the screen.

```
Enter name of 2D node specifications file:
Enter name of 2D averaging function specification file:
```

Many of FIELDGEN2D_SVA's remaining prompts are similar to those of FIELDGEN_SVA and FIELDGEN3D_SVA. Its next prompt is:

```
Express stochasticity in natural or log domain? [n/l]:
```

If your answer to the first of the above prompts is "n" then stochastic fields which are generated by FIELDGEN2D_SVA are added to the mean value field that is supplied in the averaging function specification file. Alternatively, if the response is "L", then mean cell values are multiplied by the log (to base 10) of FIELDGEN2D_SVA-generated stochastic fields. Variances supplied in the averaging function specification file must therefore pertain to the log of the hydraulic property for which stochastic fields are being generated. (Note, however, that this does not apply to the mean.)

The nature of the moving average function is next requested:

```
Is moving ave spherical, exponential, gaussian or power? [s/x/g/p(..)]:
```

See documentation for FIELDGEN_SVA. It is important that the moving average function be distinguished from a variogram. In theory, the stochastic field that results from application of a moving averaging function can be described by a (spatially varying) variogram. However, for all but the Gaussian averaging function, this variogram does not have a compact analytical expression. Its range is of the order of the square root of 2 times that of the averaging function.

FIELDGEN2D_SVA next asks for the number of realisations that it must generate. Be careful in answering this question. This is because FIELDGEN2D_SVA stores the stochastic fields which it generates in separate columns of a CSV file. Hence it

maintains all realisations in memory until it writes this file. FIELDGEN2D_SVA may run out of memory if a model is large and you request too many stochastic fields. (In this case, run FIELDGEN2D_SVA twice or more using different random number seeds.) The prompt is:

```
How many realizations do you wish to generate?
```

The name of a CSV output file is next requested.

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

FIELDGEN2D_SVA next asks some questions that may improve execution efficiency. A response of <Enter> to any of these prompts gives FIELDGEN2D_SVA license to accept the default.

```
Enter values for following efficiency-determining variables.
    Summation factor cutoff threshold (<Enter> if 1.0E-5):
    Array processing block size (<Enter> if 5):
    Restrict averaging to active cells? (y/n) (<Enter> if y):
```

The first of the above options limits the integration area when using an exponential or Gaussian averaging function. The "summation factor cutoff threshold" is the distance where integration ceases when populating a particular grid cell with a stochastic property value. It is the value of h for which $\exp(-h/a)$ in the former case and $\exp[(-h/a)^2]$ in the latter case has the user-specified value (the default value being 1E-5).

The second prompt determines how many stochastic property arrays FIELDGEN2D_SVA fills at once. The most numerically efficient alternative is for the block size to be equal to the number of realisations. However this requires that many copies be stored of the arrays that hold independent Gaussian random variables (see documentation of FIELDGEN_SVA). Memory may become an issue. This choice should therefore be based on the size of the model grid and your computer's memory.

FIELDGEN2D_SVA only generates stochastic field values for active model cells. However spatial averaging may be conducted over inactive model cells; this can reduce the propensity for boundary effects at cells that border inactive regions. Nevertheless, experience to date suggests that a response of "y" to the third of the above prompts works well.

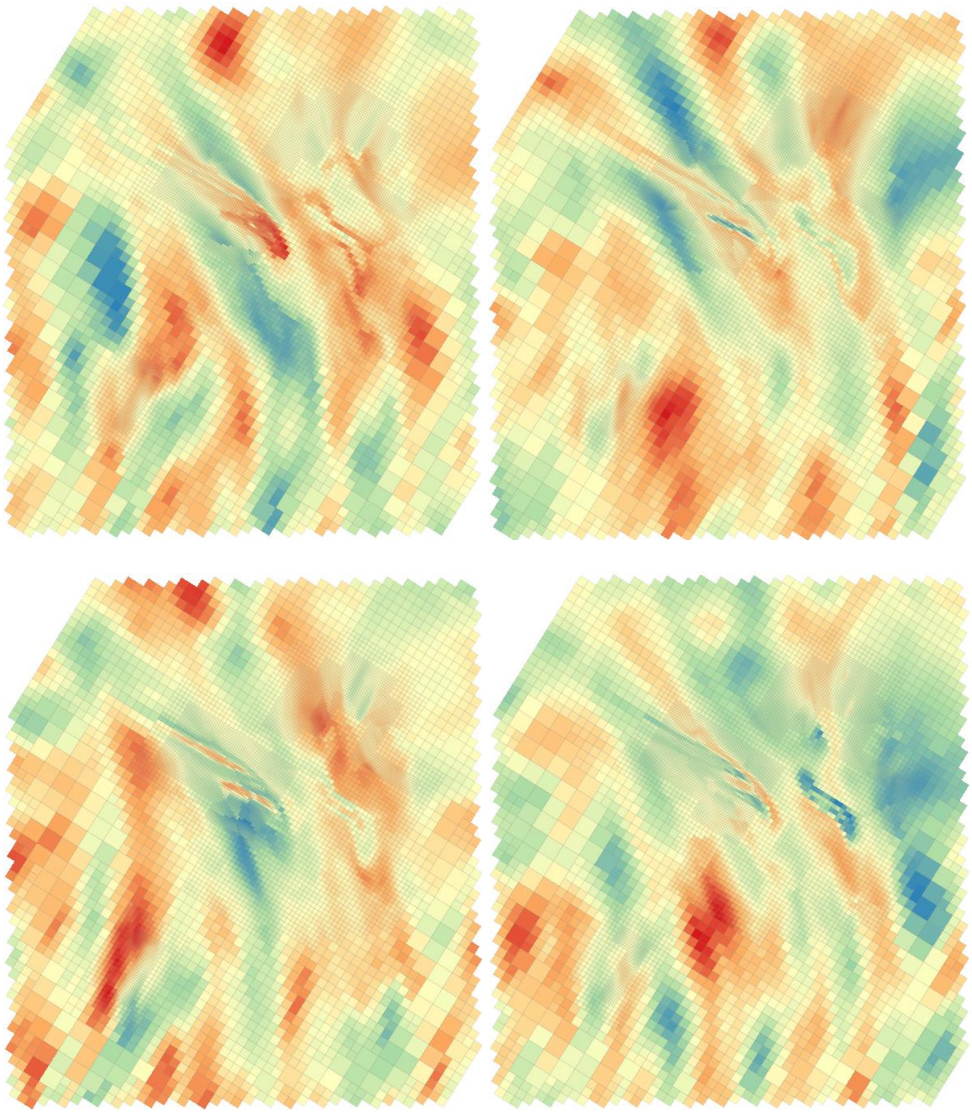
FIELDGEN2D_SVA's final prompt is:

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

Once a response to this prompt has been provided, FIELDGEN2D_SVA undertakes the calculations that are required of it, and reports its progress to the screen.

Example

The following fields populate a 2D unstructured model grid comprised of 8794 model cells per layer. The direction of anisotropy, and the anisotropy itself, vary throughout the model domain. These were interpolated to model grid cells from a series of pilot points before running FIELDGEN2D_SVA.



See Also

See also FIELDGEN, FIELDGEN_SVA, FIELDGEN2D_SVA and FILEDGEN2D_SVA1.

FIELDGEN2D_SVA1

Function of FIELDGEN2D_SVA1

As the name suggests, FIELDGEN2D_SVA1 is a variant of FIELDGEN2D_SVA. It makes use of the fact that the grid over which spatial integration of independent normal variates takes place does not have to be the same grid as that which is ultimately populated with a spatially-correlated random field. If the former is coarser than the latter, then repeated spatial integration over the former that is necessary to populate the latter can be accomplished much more quickly than if these grids are the same. The cost, however, is loss of some fine-scale stochastic detail in the fine grid.

FIELDGEN2D_SVA1 also allows conditioning of random hydraulic property fields by observed values of hydraulic properties at discrete locations.

File Types

The types of files that are read by FIELDGEN2D_SVA1 are slightly different from those that are read by FIELDGEN2D_SVA. They are tabular data files, which can also be CSV files. That is, they can be space, tab or comma-delimited.

2D Source Point File

The first of these file types is referred to as a "2D source point file". The figure below shows the first few lines of such a file displayed in Microsoft EXCEL.

x	y	AREA
467778.6	7489031	62500
469388.6	7487951	15625
470110.7	7487573	62500
472230.8	7486249	62500
467434.1	7488951	62500
469044.1	7487872	15625
469746.4	7487580	15625
471250.3	7486567	62500
466453.6	7489269	62500
468553.8	7488030	15625
468911.6	7487660	15625
469613.9	7487368	15625
470481.8	7486752	62500
465685.1	7489455	62500
467805.2	7488130	62500

First few lines of a 2D source point file.

FIELDGEN2D_SVA1 insists that certain protocols be followed by this file. The first line of a 2D source point file must contain the same headers as those that appear in the above picture. These define the contents of its respective columns. The first two columns contain the *x*, and *y* coordinates of points that will host independent Gaussian variates. These points will normally be distributed over the domain of a model grid. They may, for example, coincide with every fifth model cell. The next column contains the area associated with each of these points. This is used in spatial integration. Give every point the same (arbitrary) area if you are unsure what to do.

2D Target Point File

The first part of a 2D target point file is pictured below.

x	y	mean	sill	aa	anis	bearing
468143.1	7487845	10	0.253645	532.0665	1.034182	0.6265
468249.1	7487779	10	0.27002	658.9475	1.195867	3.334169
468421.3	7487818	10	0.289236	850.3072	1.402834	6.309551
468527.3	7487752	10	0.310618	1089.576	1.655487	9.396363
468355.1	7487712	10	0.29308	877.4917	1.446506	6.880991
468461.1	7487646	10	0.318407	1174.095	1.753477	10.46826
468633.3	7487686	10	0.331511	1373.554	1.925625	12.21391
468739.4	7487620	10	0.350402	1687.988	2.190062	14.61249
468567.1	7487580	10	0.342956	1528.817	2.083519	13.68288
468673.1	7487514	10	0.364953	1925.279	2.407049	16.37345
468845.4	7487554	10	0.363905	1931.248	2.391028	16.24897
468951.4	7487487	10	0.371539	2120.509	2.509114	17.14754
468779.1	7487447	10	0.382872	2270.309	2.690419	18.448
468885.1	7487381	10	0.399506	2641.025	2.969646	20.28862
469057.4	7487421	10	0.362305	2010.124	2.366695	16.0583

The first part of a 2D target point file.

A 2D target point file must possess seven columns. Their labels should be exactly as shown in the above figure. The contents of these columns are obvious from their labels. The points that are featured in this file will normally coincide with the centres of a structured or unstructured model grid. This is the grid that will be populated with non-stationary stochastic fields.

2D Conditioning Point File

The first part of a 2D conditioning point file is pictured below.

x	y	value
468499.3	7486498	5
468565.6	7486310	10
467419.4	7486657	15
467386.3	7486310	20
467462.5	7486078	30
467285.4	7484880	40
466377.9	7483900	50
465091.1	7482990	60

The first part of a 2D conditioning point file.

A 2D conditioning point file must possess three columns. Their labels should be exactly as shown in the above figure. The contents of these columns are obvious from their labels. The points that are featured in this file do not have to coincide with either source points or target points.

Binary IID File

This file must be written by IIDGEN. It cannot be viewed because it is binary. Information that was provided to IIDGEN when it wrote this file must be in accordance with that which is provided to FIELDGEN2D_SVA1. In particular, IIDGEN must have been informed that the number of source points for which realisations of random normal variates is required is equal to at least the number of source points provided to FIELDGEN2D_SVA1 through its 2D source point file. IIDGEN must have also been informed to provide as many random variate realisations for each source point as the number of realisations of spatially correlated target-point-associated random fields that FIELDGEN2D_SVA1 is asked to write.

Using FIELDGEN2D_SVA1

FIELDGEN2D_SVA1 commences execution by prompting for the names of the 2D source and target point files that it must read. It asks:

```
Enter name of 2D source point file:
Enter name of 2D target point file:
Enter name of 2D conditioning point file (<Enter> if none):
```

Its next prompt is:

```
Enter name of binary IID file (<Enter> if none):
```

Either press the <Enter> key, or provide the name of binary file written by program IIDGEN. This file must provide realisations of independent normal variates for at least as many source points, for at least as many realisations as the present FIELDGEN2D_SVA1 run requires. Note that if FIELDGEN2D_SVA1 is asked to produce only a single realisation of a spatially-correlated random field, then the random variates contained in this binary IID file can be accumulated over all realisations using realisation-specific factors; see documentation of IIDGEN to learn about this aspect of its functionality.

If you respond to the above prompt with <Enter>, then FIELDGEN2D_SVA1 generates realisations of random normal variates itself.

FIELDGEN2D_SVA1's following prompts are similar to those of FIELDGEN2D_SVA; see documentation of that program for details.

```
Express stochasticity in natural or log domain? [n/l]:
Is moving ave spherical, exponential, gaussian or power? [s/x/g/p(..)]:
How many realizations do you wish to generate?
```

FIELDGEN2D_SVA1's next prompts are:

```
Enter name for csv output file (<Enter> if none):
Enter name for binary output file (<Enter> if none):
```

You must ask for at least one of these output file types. Next FIELDGEN2D_SVA1 asks:

```
Enter values for following efficiency-determining variables.
   Summation factor cutoff threshold (<Enter> if 1.0E-5):
   Array processing block size (<Enter> if 5):
```

See documentation of FIELDGEN2D_SVA for the meanings of these prompts. Finally, if you have not asked FIELDGEN2D_SVA1 to obtain realisations of normal variates from a IIDGEN-generated binary file, it asks:

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

The CSV file that is written by FIELDGEN2D_SVA1 is identical to that written by FIELDGEN2D_SVA.

Conditioning

As is apparent from the above description, FIELDGEN2D_SVA1 allows conditioning of the random fields that it generates. If a target point coincides with a conditioning point, then the value of all FIELDGEN2D_SVA1-generated random fields at the target point are the conditioning value. This value also influences random values provided to nearby target points. Similarly, if a target point is close to a conditioning point, values generated for the target point are still random; however the variance associated with the target point is reduced, while random values ascribed to the target point are influenced by the conditioning point value.

The conditioning algorithm that is employed by FIELDGEN2D_SVA1 is heuristic. It is as follows.

In accordance with its random field generation algorithm, FIELDGEN2D_SVA1 ascribes an averaging function to each target point. The details of this function reflect values that are assigned to each target point through the target point file. The averaging function has a maximum value of 1.0 at the target point, this being the point on which it is centred.

Let us denote the averaging function that is associated with target point i as f_i. We denote its value at the j'th conditioning point as f_{ij}. For FIELDGEN2D_SVA1, this value can never be less than 0.0.

At each target point *i*, an effective "distance" to all conditioning points *j* is calculated as:

$$d_{ij} = (1-f_{ij})/f_{ij}$$

This distance is zero when a conditioning point coincides with a target point. The distance increases to infinity as the real distance between these two points increases. (Recall that *f* is a monotonically decreasing function of distance). A revised mean field value at the target point is then calculated using inverse-distance-weighted "spatial" interpolation from conditioning points as follows. Let:

$$w_{ij}=1/d_{ij}$$

We denote the new mean field value at target point *i* as m'_i. It is calculated from the assigned mean m_i at the target point, and conditioning values c_j assigned to conditioning points using the following equation:

$$m'_i = m_i + \frac{\sum_j (c_j - m_i) w_{ij}}{\sum_j w_{ij}}$$

A revised variance v'_i is also ascribed to the target point. This is zero where a conditioning point coincides with a target point, and is small where a conditioning point is close to a target point.

$$v'_i = v_i (1 - max_j f_{ij})$$

Calculation of the random field value at the target point through spatial averaging of independent Gaussian variates at source points then proceeds in the normal manner.

See Also

FIELDGEN2D_SVA and FIELDGEN3D_SVA1.

FIELDGEN3D_SVA

Function of FIELDGEN3D_SVA

FIELDGEN3D_SVA generates stochastic fields for a three-dimensional model using the non-centred method. The model grid can be structured or unstructured. Stationarity is not assumed. The variables which govern stochastic field variability and spatial correlation can vary on a cell-by-cell basis.

Some Theory

Theory is described in documentation of FIELDGEN_SVA. For FIELDGEN3D_SVA this theory is applied in three dimensions. FIELDGEN3D_SVA uses the same averaging functions as does FIELDGEN_SVA. However because of the extra dimension, stochastic field generation can be slow – particularly if an exponential, Gaussian or spherical averaging function is employed. Hence a power function is recommended.

The ellipsoid of anisotropy can vary on a cell-by-cell basis. Specifications for this ellipsoid requires that six variables be provided. These are now described.

The first three variables are named *a_hmax*, *a_hmin* and *a_vert*. These are the "*a*" variables that appear in equations for the averaging function in each of three perpendicular directions. It is generally assumed that the direction of maximum elongation of the anisotropy ellipsoid is close to horizontal and that, perpendicular to this, and also in a direction that is close to horizontal, is the intermediate axis of the ellipsoid. These directions are those to which *a_hmax* and *a_hmin* pertain. (Note however, that there is nothing to stop a user from making *a_hmin* greater than *a_hmax* over part or all of a model domain if he/she likes.) Perpendicular to both of these axes, and hence generally pointing roughly vertically, is the axis to which *a_vert* pertains. (The value assigned to this variable can, if a user decides, be locally greater than *a_hmax* or *a_hmin*. Also, large deviations from verticality are allowed.)

The remaining variables which govern the disposition of the anisotropy ellipsoid are angles. These define the directions of the axes of the anisotropy ellipsoid. The names of these variables are *bearing*, *dip* and *rake*.

Bearing must be between -360 degrees and 360 degrees. This is the direction in which the positive direction of the *a_hmax* axis points. It is measured from north. *Dip* is the angle in which the positive direction of the *a_hmax* axis points. Values for *dip* can vary between -180 degrees and 180 degrees. Note that, to conform with GSLIB protocols, a downwards dip is negative and an upwards dip is positive. The last variable is *rake*. Picture yourself looking down the *a_hmax* axis from its positive end towards its negative end. Now picture the ellipsoid being rotated anticlockwise about this *a_hmax* axis so that the positive direction of the *a_hmin* axis rises. The angle by which it rises is the *rake*. The value supplied for *rake* must be between -90 degrees and 90 degrees.

Note that the positive directions of each of the *a_hmax*, *a_hmin* and *a_vert* axes are assumed to form a right hand coordinate system. A screw that is turned from positive *a_hmax* to positive *a_hmin* therefore advances in the *a_vert* direction.

File Types

FIELDGEN3D_SVA requires two tabular data files. These can be CSV files if you like; that is, the contents of these files can be space or comma-delimited.

3D Node Specification File

The first of these file types is referred to as a "3D node specification file". This is not unlike a grid specification file. However specifications pertain to nodes of a model grid as cell vertices are unimportant FIELDGEN3D_SVA does not care whether the model grid is structured or unstructured.

The figure below shows the first few lines of such a node specification file displayed in Microsoft EXCEL.

node	x	y	z	layer	area	height	active
1	6.83	-1.83	-5.013	1	100	10.025	0
2	15.49	3.17	-5.037	1	100	10.075	0
3	24.151	8.17	-5.062	1	100	10.125	0
4	32.811	13.17	-5.088	1	100	10.175	1
5	41.471	18.17	-5.112	1	100	10.225	1
6	50.131	23.17	-5.138	1	100	10.275	1
7	58.792	28.17	-5.162	1	100	10.325	1
8	67.452	33.17	-5.188	1	100	10.375	1
9	76.112	38.17	-5.213	1	100	10.425	1
10	84.772	43.17	-5.237	1	100	10.475	1
11	93.433	48.17	-5.263	1	100	10.525	1
12	102.093	53.17	-5.287	1	100	10.575	1

First few lines of a 3D node specification file.

FIELDGEN3D_SVA insists that certain protocols be followed by this file. The first line of a node specification file must contain the same headers as those that appear in the above picture. These define the contents of respective columns. The first column must contain model node numbers. FIELDGEN3D_SVA insists that these be numbered sequentially starting from 1. The *x*, *y* and *z* coordinates of nodes (normally cell centres) follow in ensuing columns. The next column contains cell layer numbers while the next two columns contain cell areas and heights. The final column specifies whether a cell is active or not. FIELDGEN3D_SVA does not calculate stochastic property field value for inactive cells.

Note that cell areas and heights are used in the integration procedure through which the spatial averaging function is applied.

3D Averaging Function Specification File

The first part of a 3D averaging function specification file is pictured below.

node	mean	variance	a_hmax	a_hmin	a_vert	bearing	dip	rake
1	1	2	400	100	99.65	30	0.45	0
2	1	2	400	100	98.95	30	1.35	0
3	1	2	400	100	98.25	30	2.25	0
4	1	2	400	100	97.55	30	3.15	0
5	1	2	400	100	96.85	30	4.05	0
6	1	2	400	100	96.15	30	4.95	0
7	1	2	400	100	95.45	30	5.85	0
8	1	2	400	100	94.75	30	6.75	0
9	1	2	400	100	94.05	30	7.65	0
10	1	2	400	100	93.35	30	8.55	0
11	1	2	400	100	92.65	30	9.45	0
12	1	2	400	100	91.95	30	10.35	0
13	1	2	400	100	91.25	30	11.25	0
14	1	2	400	100	90.55	30	12.15	0
15	1	2	400	100	89.85	30	13.05	0
16	1	2	400	100	89.15	30	13.95	0
17	1	2	400	100	88.45	30	14.85	0

The first part of a 3D averaging function specification file.

A 3D averaging function specification file must possess nine columns. Their labels should be exactly as shown in the above figure. The contents of these columns are obvious from their labels. As for the 3D node specification file, the first column must feature grid nodes. These must be sequential and start at 1. All grid nodes must be cited in this column whether or not they are active.

Note that a user may wish to supply data for only active nodes of a model grid in the 3D node and averaging function specification files. Hence some nodes of a model grid may be omitted from both of these files. FIELDGEN3D_SVA does not care about this, for it has no knowledge of the grid structure. However, if a user chooses to do this then he/she must still ensure that nodes are numbered sequentially starting at 1 in both of the files that are supplied to FIELDGEN3D_SVA, and that nodes in these two files correspond to each other.

Using FIELDGEN3D_SVA

FIELDGEN3D_SVA begins execution by prompting for the names of the above two files. It then reads these files, reporting any errors that it encounters to the screen.

```
Enter name of 3D node specifications file:
Enter name of 3D averaging function specification file:
```

Its next prompt is as follows:

```
Use one-minus-Layer or eleVation as effective z coordinate? [L/V]:
```

As is apparent from the above prompt FIELDGEN3D_SVA can use cell layer numbers as pseudo *z*-coordinates. Experience in using FIELDGEN3D_SVA so far suggests that this is not a good idea. Hence a response of "v" to the above prompt is generally the best option.

Many of FIELDGEN3D_SVA's remaining prompts are similar to those of FIELDGEN_SVA. Its next prompt is:

```
Express stochasticity in natural or log domain? [n/l]:
```

If your answer to the first of the above prompts is "n" then stochastic fields which are generated by FIELDGEN3D_SVA are added to the mean value field that is supplied in the averaging function specification file. Alternatively, if the response is "L", then mean cell values are multiplied by the log (to base 10) of FIELDGEN3D_SVA-generated stochastic fields. Variances supplied in the averaging function specification file must therefore pertain to the log of the hydraulic property for which stochastic fields are being generated. (Note, however, that this does not apply to the mean.)

The nature of the moving average function is next requested:

```
Is moving ave spherical, exponential, gaussian or power? [s/x/g/p(..)]:
```

See documentation for FIELDGEN_SVA. It is important that the moving average function be distinguished from a variogram. In theory, the stochastic field that results from application of a moving averaging function can be described by a (spatially varying) variogram. However, for all but the Gaussian averaging function, this variogram does not have a compact analytical expression. Its range is of the order of the square root of 2 times that of the averaging function.

Calculation of a moving average in three dimensions can be time-consuming. By far the most efficient option is the power option, as the integration range is limited. To request a power of, say, 0.5 respond to the above prompt with "`p(0.5)`".

FIELDGEN3D_SVA next asks for the number of realisations that it must generate. Be careful in answering this question. This is because FIELDGEN3D_SVA stores the stochastic fields which it generates in separate columns of a CSV file. Hence it maintains all realisations in memory until it writes this file. FIELDGEN3D_SVA may run out of memory if a model is large and you request too many stochastic fields. (In this case, run FIELDGEN3D_SVA twice or more using different random number seeds.) The prompt is:

```
How many realizations do you wish to generate?
```

The name of a CSV output file is next requested.

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

FIELDGEN3D_SVA next asks some questions that may improve execution efficiency. A response of <Enter> to any of these prompts gives FIELDGEN3D_SVA license to accept the default.

```
Enter values for following efficiency-determining variables.
    Summation factor cutoff threshold (<Enter> if 1.0E-5):
    Array processing block size (<Enter> if 5):
    Restrict averaging to active cells? (y/n) (<Enter> if y):
```

The first of the above options limits the integration area when using an exponential or Gaussian averaging function. The "summation factor cutoff threshold" is the distance where integration ceases when populating a particular grid cell with a stochastic property value. It is the value of *h* for which exp(-*h*/*a*) in the former case and exp[(-*h*/*a*)^2] in the latter case has the user-specified value (the default value being 1E-5).

The second prompt determines how many stochastic property arrays FIELDGEN3D_SVA fills at once. The most numerically efficient alternative is for the block size to be equal to the number of realisations. However this requires that many copies be stored of the arrays that hold independent Gaussian random variables (see documentation of FIELDGEN_SVA). Memory may become an issue. This choice should therefore be based on the size of the model grid and your computer's memory.

FIELDGEN3D_SVA only generates stochastic field values for active model cells. However spatial averaging may be conducted over inactive model cells; this can reduce the propensity for boundary effects at cells that border inactive regions. Nevertheless, experience to date suggests that a response of "y" to the third of the above prompts works well.

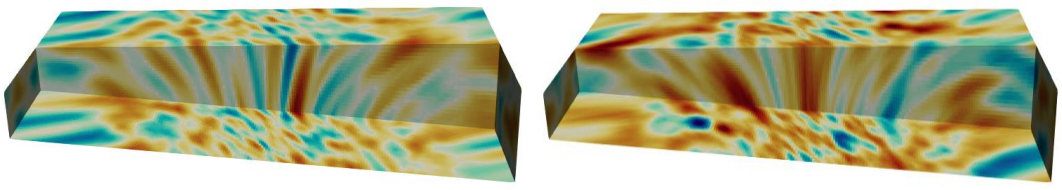
FIELDGEN3D_SVA's final prompt is:

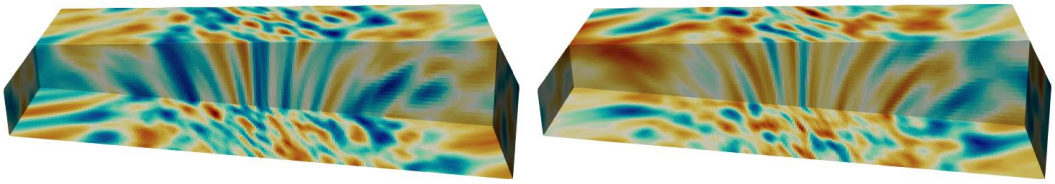
```
Enter integer seed for random number generator [324853]:
```

Once a response to this prompt has been provided, FIELDGEN3D_SVA undertakes the calculations that are required of it, and reports its progress to the screen.

Example

The following fields populate a 3D structured model grid with 100 rows, 200 columns and 20 layers. Horizontal cell widths are 10m × 10m while vertical cell widths range from 10m to 20m. *a_hmax* and *a_hmin* are uniformly 400m and 100m. *a_vert* varies between 100m and 30m in different parts of the model domain. The dip of the *a_hmax* axis varies between horizontal (at the eastern and western edges of the model domain and vertical (at the centre of the model domain). The averaging function is the power function, with a *power* value of 1.0.





See Also

See also FIELDGEN, FIELDGEN_SVA, FIELDGEN2D_SVA and FIELDGEN3D_SVA1.

FIELDGEN3D_SVA1

Function of FIELDGEN3D_SVA1

As the name suggests, FIELDGEN3D_SVA1 is a variant of FIELDGEN3D_SVA. It makes use of the fact that the grid over which spatial integration of independent normal variates takes place does not have to be the same grid as that which is populated with a spatially-correlated random field. If the former is coarser than the latter, then repeated spatial integration over the former that is necessary to populate the latter can be accomplished much more quickly than if these grids are the same. The cost, however, is loss of some fine-scale stochastic detail in the fine grid.

File Types

The types of files that are read by FIELDGEN3D_SVA1 are slightly different from those that are read by FIELDGEN3D_SVA. They are tabular data files, which can also be CSV files. That is, they can be space, tab or comma-delimited.

3D Source Point File

The first of these file types is referred to as a "3D source point file". The figure below shows the first few lines of such a file displayed in Microsoft EXCEL.

x	y	z	vol
41.471	18.17	-5.112	1022.5
84.772	43.17	-5.237	1047.5
128.074	68.17	-5.362	1072.5
171.375	93.17	-5.487	1097.5
214.676	118.17	-5.612	1122.5
257.977	143.17	-5.737	1147.5
301.279	168.17	-5.862	1172.5
344.58	193.17	-5.987	1197.5
387.881	218.17	-6.112	1222.5
431.183	243.17	-6.237	1247.5
474.484	268.17	-6.362	1272.5
517.785	293.17	-6.487	1297.5
561.086	318.17	-6.612	1322.5
604.388	343.17	-6.737	1347.5
647.689	368.17	-6.862	1372.5
690.99	393.17	-6.987	1397.5
734.291	418.17	-7.112	1422.5

First few lines of a 3D source point file.

FIELDGEN3D_SVA1 insists that certain protocols be followed by this file. The first line of a 3D source point file must contain the same headers as those that appear in the above picture (or close approximations to them). These define the contents of its respective columns. The first three columns contain the *x*, *y* and *z* coordinates of points that will host independent Gaussian variates. These points will normally be distributed over the domain of a model grid. They may, for example, coincide with every fifth model cell. The next column contains the volume associated with each of these points. This is used in spatial integration. Give every point the same (arbitrary) volume if you are unsure what to do.

3D Target Point File

The first part of a 3D target point file is pictured below.

x	y	z	mean	sill	a_hmax	a_hmin	a_vert	bearing	dip	rake
6.83	-1.83	-5.013	1	2	400	100	99.65	30	0.45	0
15.49	3.17	-5.037	1	2	400	100	98.95	30	1.35	0
24.151	8.17	-5.062	1	2	400	100	98.25	30	2.25	0
32.811	13.17	-5.088	1	2	400	100	97.55	30	3.15	0
41.471	18.17	-5.112	1	2	400	100	96.85	30	4.05	0
50.131	23.17	-5.138	1	2	400	100	96.15	30	4.95	0
58.792	28.17	-5.162	1	2	400	100	95.45	30	5.85	0
67.452	33.17	-5.188	1	2	400	100	94.75	30	6.75	0
76.112	38.17	-5.213	1	2	400	100	94.05	30	7.65	0
84.772	43.17	-5.237	1	2	400	100	93.35	30	8.55	0
93.433	48.17	-5.263	1	2	400	100	92.65	30	9.45	0
102.093	53.17	-5.287	1	2	400	100	91.95	30	10.35	0
110.753	58.17	-5.312	1	2	400	100	91.25	30	11.25	0
119.413	63.17	-5.338	1	2	400	100	90.55	30	12.15	0
128.074	68.17	-5.362	1	2	400	100	89.85	30	13.05	0
136.734	73.17	-5.388	1	2	400	100	89.15	30	13.95	0

The first part of a 3D target point file.

A 3D target point file must possess eleven columns. Their labels should be as shown in the above figure (or approximations to them). The contents of these columns are obvious from their labels. The points that are featured in this file will normally coincide with the centres of a structured or unstructured model grid. This is the grid that will be populated with non-stationary stochastic fields.

Note that if a sill (or variance) of a target point is negative, then this informs FIELDGEN3D_SVA1 that the corresponding target point should not be populated with random numbers. Instead, a value of 1.1D31 is assigned to that point. (This allows inactive grid cells to feature in the target point file, and hence in FIELDGEN3D_SVA1 output files, while FIELDGEN3D_SVA1 execution is not slowed down by having to populate these cells with random numbers.)

3D Conditioning Point File

The first part of a 3D conditioning point file is pictured below.

x	y	z	value
778.048	62.382	-6.763	4.21
1632.83	301.856	-8.912	2.43
645.657	-568.311	-5.688	8.29
979.554	213.362	-22.162	2.43
653.958	-482.689	-17.438	5.21
1732.996	648.362	-47.812	2.12
787.619	-694.195	-29.188	4.12
985.727	-337.33	-46.987	2.83
1629.528	207.574	-79.088	1.23

The first part of a 3D conditioning point file.

A 3D conditioning point file must possess four columns. Their labels should approximate those shown in the above figure. The contents of these columns are obvious from their labels. The points that are featured in this file do not have to coincide with either source points or target points.

Binary IID File

This file must be written by IIDGEN. It cannot be viewed because it is binary. Information that was provided to IIDGEN when it wrote this file must be in accordance with that which is provided to FIELDGEN3D_SVA1. In particular, IIDGEN must have been informed that the number of source points for which realisations of random normal variates is required is equal to at least the number of source points provided to FIELDGEN3D_SVA1 through its 3D source point file. IIDGEN must have also been informed to provide as many random variate realisations for each source point as the number of realisations of spatially correlated target-point-associated random fields that FIELDGEN3D_SVA1 is asked to write.

Using FIELDGEN3D_SVA1

FIELDGEN3D_SVA1 commences execution by prompting for the names of the 3D source and target point files that it must read. It asks:

```
Enter name of 3D source point file:
Enter name of 3D target point file:
Enter name of 3D conditioning point file (<Enter> if none):
```

Its next prompt is:

```
Enter name of binary IID file (<Enter> if none):
```

Either press the <Enter> key, or provide the name of binary file written by program IIDGEN. This file must provide realisations of independent normal variates for at least as many target points, for at least as many realisations as the present FIELDGEN3D_SVA1 run requires. Note that if FIELDGEN3D_SVA1 is asked to produce only a single realisation of a spatially-correlated random field, then the random variates contained in this binary IID file can be accumulated over all

realisations using realisation-specific factors; see documentation of IIDGEN to learn about this aspect of its functionality.

If you respond to the above prompt with <Enter>, then FIELDGEN3D_SVA1 generates realisations of random normal variates itself.

Its following prompts are similar to those of FIELDGEN3D_SVA; see documentation of that program for details.

```
Express stochasticity in natural or log domain? [n/l]:
Is moving ave spherical, exponential, gaussian or power? [s/x/g/p(..)]:
How many realizations do you wish to generate?
```

Next, FIELDGEN3D_SVA1 prompts for the names of the files that it must write:

```
Enter name for csv output file (<Enter> if none):
Enter name for binary output file (<Enter> if none):
```

You must ask for at least one of these output files. Note, however, that if you provide the name of a CSV file then FIELDGEN3D_SVA1 will ask you if you need a header in this CSV file. The prompt is:

```
Include header in this file? [y/n]:
```

If a CSV file does not have a header, and if it is comprised of only a single column, then it can serve as an input file for models such as MODFLOW.

FIELDGEN3D_SVA1's next prompts are similar to those of FIELDGEN3D_SVA:

```
Enter values for following efficiency-determining variables.
   Summation factor cutoff threshold (<Enter> if 1.0E-5):
   Array processing block size (<Enter> if 5):
```

Finally, if you have not asked to read random realisations of normal variates from an IIDGEN-generated binary file, FIELDGEN3D_SVA1 asks:

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

The CSV file that is written by FIELDGEN3D_SVA1 is identical to that written by FIELDGEN3D_SVA, except that it is missing the initial column in which nodes are numbered sequentially.

Conditioning

For a description of the conditioning algorithm see documentation of FIELDGEN2D_SVA1.

If undertaking conditioning using a Gaussian or exponential averaging function it may be a good idea to set the simulation factor cutoff threshold to a very low value. This can prevent lines of discontinuity in random parameter field realisations.

Inactive Target Points

Recall from the specifications of a target point file that if the variance associated with a target point is negative, then all "random" values ascribed to that point are 1.1E31.

Alternatively, it its variance is zero, then all random values ascribed to that point are the same, these being its user-supplied mean value.

See Also

FIELDGEN3D_SVA, FIELDGEN2D_SVA and FIELDGEN2D_SVA2.

FIELDGEN3D_SVA2

Function of FIELDGEN3D_SVA2

FIELDGEN3D_SVA2 is slightly modified from FIELDGEN3D_SVA1. However its numerical algorithm is the same.

The primary purpose of FIELDGEN3D_SVA2 is not to generate a suite of random fields for the use of a model. Rather, its task is to create a single model hydraulic property field based on a single set of independent normal variates pertaining to source points. These variates may be undergoing estimation by PEST or PEST++.

Hence the task that is performed by FIELDGEN3D_SVA2 resembles interpolation from a dense array of pilot points under conditions where stochasticity is spatially variant.

Using FIELDGEN3D_SVA2

Many of the prompts issued by FIELDGEN3D_SVA2 are the same as those issued by FIELDGEN3D_SVA1. As usual, by responding to any prompt with “e”, you can be taken back to the previous prompt.

Its first three prompts are as follows:

```
Enter name of 3D source point file:
Enter name of 3D target point file:
Enter name of 3D conditioning point file (<Enter> if none):
```

See documentation of FIELDGEN3D_SVA1 for the roles and formatting of these files. Next FIELDGEN3D_SVA2 asks:

```
Enter name of ascii IID file:
Does this file have a header (y/n):
```

The ASCII IID file must contain a single column of numbers (with an optional header). There must be as many entries in this file as there are 3D source coordinates. Each entry associates a variable with a source point. If working with PEST or PEST++ there will probably be a template for this file.

The next two prompts replicate those of FIELDGEN3D_SVA2:

```
Express stochasticity in natural or log domain? [n/l]:
Is moving ave spherical, exponential, gaussian or power? [s/x/g/p(..)]:
```

FIELDGEN3D_SVA2 next asks:

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

This text file will contain a single column, with no header. It will contain as many lines as the target point file. Under some circumstances, this can serve as a model input file.

FIELDGEN3D_SVA2's final prompt is identical to one of FIELDGEN3D_SVA1's final prompts:

```
Enter summation factor cutoff threshold (<Enter> if 1.0E-5):
```

FIELDGEN3D_SVA2 next undertakes spatial integration, records its output file, and then ceases execution.

Using FIELDGEN3D_SVA2

If there are as many target points as there are nodes in a model grid, the FIELDGEN3D_SVA2 output file may be directly readable by the model (for example using MODFLOW's OPEN/CLOSE functionality). However it is possible that the model possesses more cells than there are target points. This may happen if 3D hydraulic property construction is undertaken for only a part of the model domain. Under these circumstances, some post-processing of the FIELDGEN3D_SVA2 output file will be required before use by the model.

See Also

See other programs from the FIELDGEN3D_SVA* family.

GENREAL2SRF

Function of GENREAL2SRF

GENREAL2SRF writes a SURFER grid file after interpolating from the cell centres of a MODFLOW real array to the nodes of the SURFER grid. The MODFLOW grid on which the real array is based can be of arbitrary design, orientation and geographical location.

Using GENREAL2SRF

Like other programs of the Groundwater Data Utility suite, GENREAL2SRF checks for the presence of a settings file *setting.fig* on commencement of execution. It reads the COLROW specification from this file.

Next it prompts for the name of the grid specification file. From this file it reads the location and specifications of the MODFLOW grid.

Its next prompts are:

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

As the name suggests, the purpose of the window integer array file is to indicate inclusion or exclusion of various parts of the real array from interpolation to the SURFER grid. Any real array cell for which the corresponding integer array element is zero is not interpolated to the SURFER grid.

Interpolation to the SURFER grid is bilinear. Thus each SURFER grid node is informed by a maximum of four MODFLOW cell centres, namely those that surround it. However if a SURFER grid node lies outside the MODFLOW grid, or outside of a cell with non-zero window integer array value, then that node is blanked.

Next GENREAL2SRF asks for details of the SURFER grid to which interpolation must take place. Its prompts are:

```
Enter specifications for SURFER grid:
   X direction grid minimum:
   X direction spacing:
   No. of X direction nodes:

   Y direction grid minimum:
   Y direction spacing:
   No. of Y direction nodes:
```

Finally it asks for the name of the SURFER grid file which it must write:

```
Enter name for SURFER grid file:
```

Uses of GENREAL2SRF

GENREAL2SRF is more flexible than REAL2SRF in that the latter translates a MODFLOW real array to SURFER grid file format only if the MODFLOW grid on which the array is based has uniform dimensions. This translation involves no interpolation - just a re-writing of the array in different format.

GENREAL2SRF is different, however. The MODFLOW and SURFER grids do not need to coincide. Furthermore, transferral of values from one to the other requires spatial interpolation. Hence the SURFER grid file does not constitute an exact reproduction of information within the MODFLOW grid file. However use of GENREAL2SRF is far more flexible than that of REAL2SRF.

See Also

See also REAL2SRF.

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 whether 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	"uniform" or "file"

	equation is obtained	
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 file 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 whether 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 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	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 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	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 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	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.

GETMULARR

Function of GETMULARR

GETMULARR performs a function that is somewhat similar to that of the MANY2ONE utility in that it extracts individual arrays from MODFLOW and MT3D unformatted output files. However where the latter files are large because they hold many arrays, use of MANY2ONE becomes cumbersome as it presents the contents of each array to the user, asking him/her whether it is his/her desire that the array be stored. In contrast to this, GETMULARR stores arrays corresponding to simulation times and layers that are pre-set by the user. The times and layers for which arrays are recorded in a MODFLOW/MT3D output file can be acquired through the prior running of the ARRDET utility.

Using GETMULARR

Immediately upon commencement of execution, GETMULARR checks for the presence of a settings file *settings.fig*. If this file is not present in the directory from which it is run, GETMULARR ceases execution with an error message. It reads from this file the protocol for storage of formatted real arrays. GETMULARR needs to know this because it writes extracted arrays in formatted form. In particular, if the COLROW variable in *settings.fig* is set to "yes", then a number-of-columns, number-of-rows header comprises the first line of any formatted real array file which GETMULARR writes. Alternatively, if it is set to "no", this header is omitted.

GETMULARR's first prompt is:-

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

If a "filenames file" named *files.fig* is present in the directory from which it is run, and if this file contains the name of a grid specification file, then the name of the latter will be included in the above prompt as its default response.

Next GETMULARR asks:-

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

This file may have been written by MODFLOW (in which case it is a head or drawdown file) or by MT3DMS (in which case it is a concentration file). In each case the file is presumed to be a binary file in which pertinent arrays are stored layer by layer. Each layer is preceded by a header which provides (together with other data) the layer number and total simulation time to which the array pertains.

GETMULARR's next prompt is:-

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

Enter "f" or "t" as appropriate. (GETMULARR needs to know the file type as MODFLOW and MT3DMS employ slightly different protocols for array headers.)

Then GETMULARR asks:-

```
 Enter name of array extraction file:
```

The array extraction file must be prepared by the user prior to running GETMULARR. An example of an array extraction file follows.

```
  300.0000            1     head1.ref
#  300.0000           2     head2.ref

  300.0000            3     head3.ref
  300.0000            4    "he ad4.ref"
  300.0000           15     head15.dat
```

Part of an array extraction file.

Any line of an array extraction file in which the first non-blank character is "#"is ignored. Blank lines are also ignored. All other lines must have three entries, the first of which is a simulation time (i.e. the MODFLOW/MT3DMS "totim" variable. The second entry on each line is a MODFLOW/MT3DMS layer number. The third entry is the name of a file. GETMULARR writes the array that it extracts for the nominated layer at the pertinent simulation time to the nominated file. It writes the file in ASCII format (with or without a number-of-columns, number-of-rows header as described above).

If GETMULARR cannot find an array for a time and layer specified in an array extraction file, it ceases execution with an appropriate error message.

It is important to note that times and layers supplied in the array extraction file must be provided in the same order as that in which arrays are stored in the MODFLOW/MT3D unformatted output file. This is in order of increasing time and layer number.

Uses of GETMULARR

An array extraction file suitable for the use of GETMULARR can be readily built from an ARRDET output file.

GETMULARR can be useful where multiple models are being calibrated simultaneously, with the first being a steady state model and the second being a transient version of the same model, and where outputs from the former provide initial conditions for the latter. Arrays stored at the end of unformatted heads and concentration files produced by the former model can be re-written to real arrays comprising part of the input dataset for the second model. The MOD2ARRAY utility can assist in preparation of such a dataset.

See Also

See also ARRDET, GETMULARR1, MOD2ARRAY.

GETMULARR1

Function of GETMULARR1

GETMULARR1 reads an unformatted MODFLOW or MT3D heads or concentration output file. It writes all arrays found in that file corresponding to a user-specified simulation time to another unformatted file in identical format. The new file thus emulates a MODFLOW/MT3D heads/concentration output file in which heads or concentration arrays were recorded at only one output time.

Using GETMULARR1

GETMULARR1's first prompt is:-

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

If a "filenames file" named *files.fig* is present in the directory from which GETMULARR1 is run, and if this file contains the name of a grid specification file, then the name of the latter will be included in the above prompt as its default response.

Next GETMULARR1 asks:-

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

This file may have been written by MODFLOW (in which case it is a head or drawdown file) or by MT3DMS (in which case it is a concentration file). In each case the file is presumed to be a binary file in which pertinent arrays are stored layer by layer. Each layer is preceded by a header which provides (together with other data) the layer number and total simulation time to which the array pertains.

GETMULARR1's next prompt is:-

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

Enter "f" or "t" as appropriate. (GETMULARR1 needs to know the file type as MODFLOW and MT3DMS employ slightly different protocols for array headers.)

Then GETMULARR1 asks:-

```
Enter simulation time for which to extract arrays:
```

If you are unsure what simulation times are represented in a MODFLOW/MT3D unformatted output file, use the ARRDET utility; see the "total_time" column in this file.

Finally GETMULARR1 asks:-

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

Provide the name of the pseudo-MODFLOW/MT3D unformatted output file to which you would like the extracted arrays written.

GETMULARR1 then reads the nominated MODFLOW/MT3D unformatted output file and writes one of reduced size, containing only arrays pertaining to the nominated simulation time.

Uses of GETMULARR1

Sometimes it is convenient to have MODFLOW/MT3D output arrays pertaining only to a time of interest stored in MODFLOW/MT3D unformatted output files. This can be particularly useful where such files are used for obtaining initial heads or concentrations for a following simulation. The arrays of interest may lie at the end of a sequence of other arrays in the existing file. By placing these arrays at the front of a new unformatted file, MODFLOW or MT3D can be directed to read the arrays from that file when undertaking its next simulation.

See Also

See also GETMULARR, ARRDET.

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 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 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 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 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.

HANIS2BLN

Function of HANIS2BLN

HANIS2BLN reads a file in which data is arranged in columns. Column headers are optional. Each row of this file should contain (at least) the coordinates of a point, together with the correlation length, bearing (with respect to north) of anisotropy and ratio of anisotropy at that point. HANIS2BLN records a BLN file (which can be read by SURFER). This file enables plotting of correlation lengths in the major and minor directions of anisotropy.

Using HANIS2BLN

HANIS2BLN commences execution by asking for the name of the file that it must read:

```
Enter name of anisotropy specification file:
```

An example of the first part of such a file is shown below. Columns can be arranged in any order; the file may possess redundant columns.

point	x	y	kh	a	hanis	bearing
b1	10764.217	20253.609	10.5	120.0	2.5	21.0
b2	10673.250	19993.704	9.0	120.0	2.5	31.0
b3	10538.966	19751.126	8.3	120.0	2.5	26.0
b4	10491.317	19447.904	7.2	120.0	2.5	17.0
b5	10409.014	19205.326	6.5	120.0	2.5	0.0
b6	10456.663	18962.749	6.5	120.0	2.5	-43.0
etc						

HANIS2BLN (the leading "H" stands for "horizontal") next asks for the columns from which it should read pertinent information. The prompts, together with appropriate responses for reading the above file, are as follows:

```
Enter column number to read point x coordinate: 2
Enter column number to read point y coordinate: 3
Enter column number to read correlation length: 5
Enter column number to read anisotropy ratio: 6
Enter column number to read anisotropy bearing: 7
Skip a line at top of file? (y/n): y
```

Finally HANIS2BLN prompts for the name of the BLN file that is must write.

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

It then proceeds to write the file.

Used of HANIS2BLN

Many of the functions in PLPROC support interpolation based on spatially-varying anisotropy. Details of correlation lengths and anisotropy specifications are interpolated from a set of points to the entirety of the interpolation domain. It is often

instructive to plot the points from which interpolation takes place, together with the axes of the anisotropy ellipses at these points.

Program HANIS2BLN may also provide useful plotting functionality to accompany use of programs such as FIELDGEN2D_SVA1 which allow stochastic field generation based on spatially varying anisotropy.

See Also

See also PLPROC and FIELDGEN2D_SVA1.

IIDGEN

Function of IIDGEN

IIDGEN generates an array of random numbers ("iid" stands for independent and identically-distributed"). This array has *N* rows and *M* columns. The *N* rows correspond to *N* source points while the *M* columns pertain to *M* realisations. Each of these *N*×*M* random numbers is sampled from a standard normal distribution. That is, each is sampled from a distribution that has a mean of zero and a standard deviation of one. Optionally, these random numbers can be multiplied by realisation-specific factors and added to form a single column of numbers, with each accumulated number in the single column corresponding to a single source point.

(Accumulated) random numbers produced by IIDGEN can be used by programs FIELDGEN2D_SVA1 and FIELDGEN3D_SVA1.

Using IIDGEN

IIDGEN begins execution with the prompt:

```
Enter number of source points:
```

For programs FIELDGEN2D_SVA1 and FIELDGEN3D_SVA1, these are points over which spatial averaging takes place. Each of these points has a location in space. Spatial averaging of standard normal variates that are associated with these points produces parameter fields that show spatial correlation. There is no need for IIDGEN to know the locations of these source points.

Next IIDGEN asks for the number of realisations of random normal variates that are required for each source point. The prompt is:

```
Enter number of realizations:
```

Next IIDGEN asks whether each realisation should be multiplied by a realisation-specific factor, and random numbers summed for each source point. Its question is:

```
Combine realizations using factors? [y/n]:
```

If your answer to this question is "y", then IIDGEN prompts for the name of a factor file. This file can be space, tab or comma-delimited. It must contain two columns, and *M* rows, where *M* is the number of realisations. The first part of such a file (displayed in Microsoft EXCEL) is pictured below.

realisation	factor
1	1.2
2	0.4
3	-1.3
4	-0.4
5	2.3
6	1.1
7	0.89
8	-0.23
9	-0.34

The first part of a realisation factor file.

The two columns of a realisation factor file must have headers that are exactly as shown above. (However you can spell "realization" with a "z" if you wish.) A factor must be provided for each realization; surplus rows are ignored.

IIDGEN can store its array of random numbers in either or both of two file types. It prompts for the names of these files.

```
Enter name for csv output file (<Enter> if none):
Enter name for binary output file (<Enter> if none):
```

Finally, it asks for a seed for the random number generator.

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

After having received the above information, IIDGEN generates the random numbers and writes the files.

If random numbers are not summed, then the number of rows in the CSV file that IIDGEN writes is equal to the number of source points, while the number of columns is equal to the number of realisations plus one; the first column contains source point numbers (source points are referred to as "nodes" in this file). These numbers start at 1 and are incremented by 1. However if random numbers are multiplied by realisation-specific factors and added, then the number of columns in the IIDGEN-generated CSV file is just two. The first column contains source point numbers while the second column contains summed random numbers for each source point over all realisations.

The opposite protocol is adopted in recording a binary file; this protocol is a little more convenient for most of the programs which read it. If random number summation is not performed, then this file contains a header followed by *M* records. Each of its *M* records contains *N* numbers. *M* is the number of realisations and *N* is the number of source points. Where factor-based accumulation takes place, then the file has only one record, this containing accumulated random numbers over all realisations for all source points.

See Also

See also FIELDGEN2D_SVA1 and FIELDGEN3D_SVA1.

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 such as QGIS) 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 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:
```

These files should have the same filename base. They should have extensions of "*.mif*" and "*.mid*" respectively.

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.

When you import the *mif/mid* file pair into a GIS, it will ask you for the earth grid projection system that you are using. Presumably, the coordinates that are used in the model grid specification file use this same projection.

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 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 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 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 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.

LOGARRAY

Function of LOGARRAY

LOGARRAY reads a real array. It evaluates the log (to base 10) of all elements in that array. It then writes another array. This can be useful for display purposes, for example when contouring hydraulic properties such as hydraulic conductivity. If log transformation is not undertaken before contouring, details of spatial variation in the hydraulic property in areas of low property value may be lost.

Using LOGARRAY

Like many of the programs of the Groundwater Data Utilities, LOGARRAY commences execution by reading a grid specification file for the current model, from which it obtains the row and column dimensions of the finite difference grid. Note that a settings file (named *setting.fig*) must reside in the directory from which LOGARRAY is run.

LOGARRAY next prompts for the name of a real array file. After reading the array it prompts:-

```
Enter inactive threshold for array (press <ENTER> if none):
```

If the absolute value of any array element is above this threshold, that array element is left untouched by LOGARRAY. Otherwise its log is taken. However if the element is zero or negative (and its absolute value is below the threshold) an error condition is reported.

Finally LOGARRAY prompts for the name of an output file, to which it writes the log-transformed real array.

Uses of LOGARRAY

As mentioned above, LOGARRAY can be useful in re-writing arrays prior to contouring for display purposes.

See Also

See also REAL2SRF.

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 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.

See Also

See also GETMULARR and ARRDET.

MF2VTK

Function of MF2VTK

MF2VTK writes a "legacy VTK" file. This can be read by 3D visualization packages such as PARAVIEW. The VTK file that is written by MF2VTK contains the geometry of a MODFLOW grid. Optionally, it can contain user-provided MODFLOW-compatible integer and real cell-by-cell data. Hence zonation schemes can be displayed in three dimensions, as well as model hydraulic properties such as hydraulic conductivity, and/or model-calculated system states such as head and contaminant concentration.

Using MF2VTK

Like all programs of the Groundwater Data Utilities Suite, you can backtrack to a previous prompt by responding to a current prompt with "e" or "E" (for escape).

MF2VTK begins execution with the prompt:

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

As is discussed in Part A of this manual, the grid specification file informs a program of the number of rows and columns in a finite difference grid. It also provides cell dimensions. However, it provides no information on the vertical disposition of the model grid. Nor does it record which cells are active and inactive. (Only information pertaining to active model cells is recorded in the VTK file which MF2VTK writes.)

So MF2VTK next asks:

```
Enter number of layers in model:
```

Followed by:

```
Enter filename base of activity arrays:
```

and then:

```
Enter filename base of layer bottom elevation arrays:
```

An activity (i.e. IBOUND) array is required for each model layer. Suppose that you provide a filename base of *ibound* in response to the first of the above prompts, and that the model has 3 layers. Then MF2VTK expects to read files *ibound1.inf*, *ibound2.inf* and *ibound3.inf*. MF2VTK adds the layer number and "*.inf*" extension itself. You can extract these arrays from the BASIC package input file of your model using a text editor. Alternatively, the MOD2ARRAY utility documented herein can do this automatically.

A similar protocol applies to the reading of layer bottom elevation arrays. However there are a couple of important differences. Suppose that you provide a filename base of *bottom* in response to the above prompt. Then MF2VTK expects to read files named *bottom0.ref, bottom1.ref, bottom2.ref* and *bottom3.ref*. Note that the extension is now "*.ref*" (in accordance with the Groundwater Utility protocol for reading MODFLOW-compatible arrays which contain real data). Note also, that an array with a layer index of 0 is read; this is the elevation of the top of the model. All of these arrays can be extracted from the discretization file pertaining to the current model (perhaps with a little help from the MOD2ARRAY utility).

Next MF2VTK prompts for the name of the VTK file which it must write:

```
Enter name for VTK file:
```

Ensure that this file has an extension of "*.vtk*". It then asks:

```
Record scalar data in VTK file?  [y/n]:
```

If you answer "n" to this prompt, MF2VTK writes the requested VTK file, and then ceases execution. This file records the vertices of all active cells, together with some model-pertinent data, the latter facilitating display, colouration and sectioning in PARAVIEW. Recorded data are:

- Cell IBOUND values;
- Cell layer, row and column numbers.

Other cell-based data be recorded in the VTK file which MF2VTK writes if the response to the above prompt is "y". In this case MF2VTK asks:

```
Enter filename base of layer data arrays (<Enter> if no more):
Is this data integer or real [i/r]:
Enter label for this data:
```

The protocol for the reading of data arrays is similar to that described above for activity and bottom elevation arrays. MF2VTK expects to read NLAY model-compatible, two-dimensional arrays, where NLAY is the number of model layers. Suppose that the filename base for these array files is *base*. For integer data, MF2VTK expects to read files *base1.inf, base2.inf,* etc. For real data, MF2VTK expects to read files *base1.ref, base2.ref,* etc. These data are then recorded in the VTK file that MF2VTK writes (but only for active model cells) together with the accompanying label (which must be 20 characters or less in length).

It is important to note that the protocol for MODFLOW-compatible arrays that is specified in file *settings.fig* must be observed. If this file contains the line "colrow=yes", then any array file that is supplied to MF2VTK must have an NCOL NROW header line. Alternatively, if this is not the case, then any array file that is supplied to MF2VTK must not possess this line.

Uses of MF2VTK

Obviously, MF2VTK's reason for existence is to enable the display of MODFLOW model data in three dimensions. Inspecting model data in three dimensions is much more satisfying than inspecting it in two dimensions. This applies not just to the grid, but to model properties (which may have been estimated by PEST or PEST++), and to model-calculated system states. Arrays containing model-calculated system states can be extracted from a MODFLOW or MT3G-generated binary file using utilities such as MANY2ONE and GETMULARR.

See Also

See also MF2VTK1, MOD2ARRAY and GETMULARR.

MF2VTK1

Function of MF2VTK1

MF2VTK1 performs a similar role to that of MF2VTK. It records MODFLOW-related data in a "legacy VTK" file so that these data can be visualized using programs such as PARAVIEW. However, in contrast to MF2VTK, it does not read these data from a sequence of arrays. Instead, it reads them from tables in which cells of a model grid are identified by layer, row and column number. These tables may comprise part of the input dataset of a MODFLOW package such as its DRAIN or GHB package. Alternatively, these files may be specially prepared by the user. In either case, a table does not need to cite every cell in the grid of a MODFLOW model; MF2VTK1 supplies values for missing cells.

Using MF2VTK1

The questions asked by MF2VTK1 are identical to those asked by MF2VTK – up until those which follow this prompt:

```
Record scalar data in VTK file?  [y/n]:
```

If the response to this question is "y", the prompts which follow are almost identical to those which are issued by the MF62VTK1A utility. The next prompt is:

```
Enter name of tabular data file (<Enter> if no more):
```

MF2VTK1 can read data from one or a number of tabular data files. An example of the first part of such a file is provided below.

```
# MODFLOW2000 Constant-Head Boundary Package (CHD)
    16 AUXILIARY IFACE NOPRINT # MXACTC Option
    14     0 # Data Set 5: ITMP NP Stress period 1
     1     5    24  8.40000E+01   8.40000E+001      101  -101    # Data Set 6:
     1     6    45  1.11000E+02   1.10000E+002      102  -102    # Data Set 6:
     1    21    82  9.20000E+01   9.20000E+001      103  -103    # Data Set 6:
     1    26   115  1.01000E+02   1.00000E+002      104  -104    # Data Set 6:
     1    32   141  1.22000E+02   1.20000E+002      105  -105    # Data Set 6:
     1   199   198  2.40000E+02   2.40000E+002      106  -106    # Data Set 6:
     1   146   248  3.01000E+02   3.00000E+002      107  -107    # Data Set 6:
     2     5    24  8.40000E+01   8.40000E+001      108  -108    # Data Set 6:
     2     6    45  7.10000E+01   7.10000E+001      109  -109    # Data Set 6:
     2    21    82  9.20000E+01   9.20000E+001      110  -110    # Data Set 6:
     2    26   115  1.01000E+02   1.00000E+002      111  -111    # Data Set 6:
     2    32   141  1.22000E+02   1.20000E+002      112  -112    # Data Set 6:
     2   199   198  2.40000E+02   2.40000E+002      113  -113    # Data Set 6:
     2   146   248  3.01000E+02   3.00000E+002      114  -114    # Data Set 6:
```

Part of a tabular data file that can be read by MF2VTK1.

The table which MF2VTK1 must read can be preceded by other information; it can even be preceded by other tables. The table itself must be recognizable by text which is provided on the first non-blank line to precede it. The table can be terminated by either the end of the file, or by another non-blank line that can be identified using a text string.

On each line of the table that is read by MF2VTK1, three columns must be dedicated to identification of a cell. In the above example (and in most cases) these will be the first three columns of the table. However it is not essential that they be the first three columns. Nevertheless:

- The three cell-identification columns must follow each other;
- They must provide (in order), cell layer, row and column indices;
- Data that is targeted for VTK file export must reside in columns which follow the cell identification columns.

After the name of a tabular input file has been provided, MF2VTK1 asks:

```
Enter starting text for data table:
```

This is the text that MF2VTK1 uses to find the table in the input file. MF2VTK1 assumes that the table begins on the first non-blank line following that in which the above text is first found. For the example model input file presented above, suitable text may be “Stress period 1”. If this text string contains a space, it must be enclosed in quotes when responding to the above prompt. However MF2VTK1’s use of this text is case-insensitive. Note that if a table comprises the only contents of a file, and the table contains no headers from which text can be quoted in response to the above prompt, simply respond to the above prompt with <Enter>; MF2VTK1 then expects the table to begin at the start of the file (possibly following blank lines).

After receiving the response to this prompt, MF2VTK1 looks for the table. If it does not find it, it informs you of this, and then re-issues the above prompt. Once it locates the table, it asks:

```
Enter finishing text for data table:
```

If the end of the file marks the end of the table, respond to this prompt with <Enter>. However if other information follows this table on the file that MF2VTK1 reads, respond with an extract from the first non-blank line which follows the table. (MF2VTK1 ignores blank lines as it reads a table.)

Next MF2VTK1 asks for more information about the table. First:

```
How many columns in this table?
```

The number of columns supplied in response to this prompt should be sufficient to contain all of the information that you wish MF2VTK1 to read; MF2VTK1 does not need to be informed of columns beyond this. Thus, in the example provided above, the response may be “4” if the only data of interest are those in the fourth column of the file (i.e. the column that immediately follows the three columns that are used to identify cells). MF2VTK1 then asks:

```
In what column do cellid's begin?
```

Normally the response will be “1”. However if layer numbers are recorded in the *N*’th column, then the response must be “*N*”.

Next MF2VTK1 asks a set of questions that pertain to each column following cell identifier columns from which it may read cell information. Firstly:

```
Enter label for data in column #4 (<Enter> if ignore):
```

Respond with <Enter> or a label of 20 characters or less in length. It is good practice for this label not to include a space. If it does include a space, enclose it in quotes. This name will be associated with data read from this column on the VTK file which MF2VTK1 writes. This name will therefore be associated with these data when it is viewed using a platform such as PARAVIEW. Next:

```
Is this integer or real data?
```

Respond with “i” or “r” as appropriate. The next prompt is:

```
Enter value for missing cells:
```

This is the value which MF2VTK1 will ascribe to all cells that are not listed in the table. If the data type is real, MF2VTK1 then prompts:

```
Add or replace values for duplicated cells [a/r]:
```

Enter “a” or “r” as appropriate. If a column contains integer data, this prompt is not issued. Cell values are simply replaced if a cell is repeated.

The above set of prompts is repeated for each column up to the maximum number of columns that MF2VTK1 expects. Data that is read from user-specified columns is appended to the VTK file. The user is then asked whether another table should be read from another file. Once all desired files have been read, MF2VTK1 ceases execution.

Uses of MF2VTK1

The uses of MF2VTK1 are obvious. It provides a flexible means of facilitating three-dimensional display of MODFLOW input data.

See Also

See also MF2VTK and other VTK-pertinent programs documented herein.

MKPPSTAT

Function of MKPPSTAT

MKPPSTAT writes a pilot point statistical specification file for the use of program PPCOV_SVA. It obtains pilot point coordinates and zones from a traditional pilot points file. It then assigns variogram specifications to each pilot point. PPCOV_SVA can use these specifications to build a covariance matrix in which correlations and anisotropies vary throughout a model domain.

The pilot points statistical specification file written by MKPPSTAT assigns the same nugget, sill, anisotropy and bearing values to all pilot points, these values being 0.0, 1.0, 1.0 and 0.0 respectively. However the variogram "a" value (proportional to its range) is calculated by MKPPSTAT as being inversely proportional to local pilot point density. It is thus smaller for pilot points whose nearest neighbours are close, and greater for pilot points whose nearest neighbours are further away.

Using MKPPSTAT

MKPPSTAT commences execution by prompting for the name of a pilot points file. As is described in Part A of this manual, a pilot points file has five columns. The first four columns specify respectively the name, easting, northing and zone of each pilot point that is represented in the file. The last column specifies the value associated with each point. MKPPSTAT does not read this value. However a value must nevertheless be supplied. MKPPSTAT's prompt is:

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

MKPPSTAT must calculate a variogram "a" value for each pilot point that it finds in the pilot points file. It does this by first finding the average distance between that pilot point and the closest *N* pilot points, and by then multiplying this average distance by a factor. *N* and the factor must be provided by you in response to the following two prompts:

```
Enter no. of pilot points to compute local ave. pp. sepn.:
Enter factor of ave. separation for local variogram "a" value:
```

When computing the average distance from a certain pilot point to its *N* nearest neighbours, MKPPSTAT only considers points that lie within the same zone as the pilot point under consideration. A problem arises, however, if there is only one pilot point in a zone. If MKPPSTAT detects this occurrence, it prompts:

```
At least one zone in pilot points file contains only one pilot point.
Enter variogram "a" value for single pilot point in such a zone:
```

Finally, MKPPSTAT prompts for the name of the pilot points statistical specification file that it must write:

```
Enter name for pilot point statistical spec. file:
```

It then writes this file and ceases execution.

Uses of MKPPSTAT

In designing a parameterization scheme for a groundwater model, it is often convenient for pilot point spatial density to reflect local borehole density. Hence more pilot points are placed in some parts of the model domain than in others. This is done because increased availability of borehole data in certain areas of the model domain will probably result in increased calibration-inferred spatial heterogeneity of hydraulic properties in those areas. Subdued representation of spatial heterogeneity in other areas is not necessarily an outcome of the lack of heterogeneity in those areas – only an inability to infer it from information that is resident in the calibration dataset. Hence where computing resources limit the number of pilot points which can be employed in the model calibration process, it makes sense to place those points where they can provide receptacles for the information contained in the calibration dataset, and to reduce pilot point spatial density in those parts of the model domain where there is limited information to express.

Regularisation that is applied to pilot points of variable spatial density should accommodate these variations in density. Where a high spatial density of pilot points reflects the need to express local scale heterogeneity, the variogram used in regularisation should allow heterogeneity to exist at that scale. Conversely, where pilot points are placed further apart to reflect an inability to infer local heterogeneity (and hence their parameterization reflects a higher degree of upscaling than for densely placed pilot points), the spatial correlation length of hydraulic properties associated with these points is expected to be larger. So too will the variogram "a" value associated with them.

The pilot point statistical specification file which MKPPSTAT writes can be used by the PPCOV_SVA utility to construct a covariance matrix for use in pilot points prefer-value regularisation. It can also be used in ascribing random values to pilot points using the PEST RANDPAR or RANDPAR1 utilities. In the latter case you will need to modify the default sill value generated by MKPPSTAT.

As was stated above, MKPPSTAT does not ascribe anisotropy to pilot points. If you wish to introduce spatially varying anisotropy to the pilot point specification file built by MKPPSTAT, you must do this yourself by editing the file.

See Also

See also MKPPSTAT3D, PPCOV_SVA.

MKPPSTAT3D

Function of MKPPSTAT3D

MKPPSTAT3D performs an almost identical role to that of MKPPSTAT. However it works in three dimensions, rather than in two.

MKPPSTAT3D writes a three dimensional pilot point statistical specification file for the use of program PPCOV3D_SVA. It obtains pilot point coordinates and zones from a three dimensional pilot points file. It then assigns variogram specifications to each pilot point. PPCOV3D_SVA can use these specifications to build a covariance matrix in which correlations vary throughout a model domain.

The three dimensional pilot points statistical specification file written by MKPPSTAT3D assigns the same nugget, sill, horizontal anisotropy and bearing values to all pilot points. Values for nugget, sill and horizontal anisotropy are 0.0, 1.0, and 1.0 respectively. Values for the three angles that are used to denote the rotation of the ellipse of anisotropy are 0.0, 0.0 and 0.0; hence the ellipsoid is round in plan, with no rotation and no dip. The variogram "a_hmax" and "a_hmin" values (which are the same and are proportional to the horizontal range of the variogram) are calculated by MKPPSTAT3D as being inversely proportional to horizontal local pilot point density; these are calculated in the same way as MKPPSTAT calculates the "a" value for a two-dimensional variogram. The horizontal variogram range is thus smaller for pilot points whose nearest neighbours are horizontally close, and greater for pilot points whose nearest neighbours are horizontally further away.

The value for "a_vert" (proportional to the vertical range of the variogram) is calculated in a similar way. Thus it is proportional to the locally-averaged vertical separation of pilot points. This too is larger when pilot points have higher vertical separation than when they have lower local vertical separation.

Using MKPPSTAT3D

MKPPSTAT3D commences execution by prompting for the name of a three dimensional pilot points file. A three-dimensional pilot points file has six columns. The first five columns specify respectively the name, easting, northing, elevation and zone of each pilot point that is represented in the file. The last column specifies the value associated with each point. MKPPSTAT3D does not read this value. However a value must nevertheless be supplied. MKPPSTAT3D's prompt is:

```
Enter name of 3D pilot points file:
```

MKPPSTAT3D must calculate variogram "a_hmax" and "a_hmin" values for each pilot point that it finds in the pilot points file. These are assigned equal values for the same pilot point; hence horizontal isotropy is assumed. In a similar fashion to MKPPSTAT, MKPPSTAT3D calculates values for these quantities by first finding the average horizontal distance between a particular pilot point and the closest *N* pilot

points, and by then multiplying this average distance by a factor. *N* and the factor must be provided by you in response to the following two prompts:

```
Enter no. of pilot points to compute local ave. hor. pp. sepn.:
Enter factor of ave. hor. sepn. for local variogram hor. "a" value:
```

The horizontally closest *NV* pilot points to a particular pilot point (where *NV* is supplied by the user) are employed for calculating a "a_vert" value for the variogram ascribed to that pilot point. The average vertical absolute separation between this pilot point and its closest *NV* neighbours is calculated, and then multiplied by a user-supplied factor to obtain a_vert. Pertinent prompts are as follows:

```
Enter no. of pilot points to compute local ave. vert. pp. sepn.:
Enter factor of ave. vert. sepn. for local variogram vert. "a" value:
```

When computing the average horizontal or vertical distance from a certain pilot point to its *N* nearest neighbours, MKPPSTAT3D only considers points that lie within the same zone as the pilot point under consideration. A problem arises, however, if there is only one pilot point in a zone. If MKPPSTAT3D detects this occurrence, it prompts:

```
At least one zone in pilot points file contains only one pilot point.
Enter variogram hor. "a" value for single pilot point in such a zone:
Enter variogram vert. "a" value for single pilot point in such a zone:
```

Finally, MKPPSTAT3D prompts for the name of the pilot points statistical specification file that it must write:

```
Enter name for 3D pilot point statistical spec. file:
```

It then writes this file and ceases execution.

Uses of MKPPSTAT3D

See the discussion of uses for MKPPSTAT in documentation of this program.

The pilot point statistical specification file which MKPPSTAT3D writes can be used by the PPCOV3D_SVA utility to construct a covariance matrix for use in pilot points prefer-value regularisation. It can also be used in ascribing random values to pilot points using the PEST RANDPAR and RANDPAR1 utilities. In the latter case you will need to modify the default sill value generated by MKPPSTAT3D.

As was stated above, MKPPSTAT3D does not ascribe horizontal anisotropy to pilot points. If you wish to introduce spatially varying horizontal anisotropy to the pilot point specification file built by MKPPSTAT3D, you must do this yourself by editing the file.

See Also

See also MKPPSTAT, PPCOV3D_SVA.

MOD2ARRAY

Function of MOD2ARRAY

MOD2ARRAY can be employed as a precursor to the use of pilot point parameterisation, as well as null space Monte Carlo and many other types of analysis. Such analyses are much easier to perform when the MODFLOW and/or MT3D arrays on which these analyses are based (for example hydraulic property or layer elevation arrays) reside in files of their own, with one such file per model layer. Unfortunately however, not all MODFLOW/MT3D graphical user interfaces provide the option of writing MODFLOW/MT3D arrays to separate files in this manner. MOD2ARRAY was written to rectify this problem. It reads a MODFLOW or MT3D input file, and extracts arrays of interest from them, writing these arrays to separate files. It replaces array headers in the MODFLOW/MT3D input file from which these arrays are extracted with new headers which direct the attention of MODFLOW or MT3D to these files for the reading of the pertinent arrays.

Use of MOD2ARRAY is predicated on the assumption that the graphical user interface which wrote the MODFLOW/MT3D input dataset identifies arrays within the files comprising this dataset using easily recognised text strings. These strings are not part of the normal MODFLOW/MT3D input protocol; however they are written by most MODFLOW/MT3D file preparation programs.

Using MOD2ARRAY

MOD2ARRAY provides a high degree of flexibility in the way in which it reads and extracts arrays. Hence it provides the user with a number of options; it is important that these options be correctly understood if MODFLOW/MT3D is to read the MOD2ARRAY-modified MODFLOW/MT3D input dataset without incurring a run-time error.

In common with all programs of the Groundwater Data Utilities, MOD2ARRAY receives information from the user through a series of prompts. If the response to any of these prompts is simply “e” followed by <Enter>, MOD2ARRAY backtracks to the previous prompt. This allows quick recovery from mistaken input.

MOD2ARRAY begins execution by asking for the name of a grid specification file for the current model:

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

It reads the number of rows and columns comprising the current model grid from this file. It then asks for the name of the MODFLOW or MT3D input file from which it must extract data arrays.

```
Enter name of MODFLOW/MT3D input file:
```

In a MODFLOW or MT3D input file, each array is preceded by an array header, informing MODFLOW/MT3D how the array is to be read (or whether the array is in

fact uniform-valued and therefore does not need to be read at all). The protocol is slightly different between these two programs. At the time of writing, these differences include the following.

- MT3D does not allow the use of keywords such as "INTERNAL", "OPEN/CLOSE" etc;

- A LOCAT variable in MT3D of less than 100 indicates that the array already resides on another file. In contrast, the LOCAT variable in a MODFLOW input file indicates the unit number from which the array is to be read; this unit number may or may not be the same as that from which the file is currently being read.

- Certain array reading conventions are available in MT3D which are not available in MODFLOW (for example the zone and block reading conventions). These are indicated by LOCAT variable values of greater than 100 (but, at the time of writing, less than 104).

In order that it knows how to read arrays from the nominated MODFLOW/MT3D input file, MOD2ARRAY next asks the following question about the MODFLOW/MT3D input file which it must read:-

```
Does this use MODFLOW or MT3D array header convention?  [f/t]:
```

In general, respond with "f" if the file is a MODFLOW input file and "t" if it is an MT3D input file. Note however, that if a MODFLOW input file is being read, MOD2ARRAY assumes that a positive LOCAT variable indicates that the array is recorded just below the array header in the same file. Thus it cannot detect the presence of arrays on external files unless the OPEN/CLOSE convention is employed for this purpose.

MOD2ARRAY's next prompt is:-

```
Enter text identifier for arrays to be extracted from this file:
```

This is the string by which arrays are recognized. For example, a graphical user interface (GUI) may write the string "VERTICAL HYDRAULIC CONDUCTIVITY FOR LAYER N" (where the layer number is substituted for "N") after every occurrence of the pertinent array header on the MODFLOW LPF input file which it writes. In that case, enter the string "vertical" if this string is sufficient to distinguish it from array headers for other data types (notice case insensitivity). The same GUI may use the string "HYDRAULIC CONDUCTIVITY FOR LAYER N" for horizontal conductivity. Use of the string "hydraulic" is obviously insufficient to distinguish between the horizontal and vertical hydraulic conductivity headers. However the string " hydraulic" (with two spaces preceding the word "hydraulic") will allow these two header types to be distinguished. Note that in responding to the above prompt, it is not necessary that the string be enclosed in quotes; however if any spaces are contained within the string then the use of quotes is essential.

If MOD2ARRAY finds the user-specified string on any line of the MODFLOW/MT3D input file, it also searches for the word "layer" on that line (hence

the word "layer" should not be supplied with the string). If it finds this word, it attempts to read the layer number from the (possibly space-delimited) string immediately following it. If it is able to do this, the array is associated with that layer (and written to a file which is associated with the same layer); otherwise the array is assumed to be layer-independent and written to a layer-independent file. (The top elevation of the model is an example of a layer-independent array.)

MOD2ARRAY next asks:-

```
Are this/these integer array(s) or real array(s) [i/r]:
```

Either is permitted. Examples of integer arrays are IBOUND arrays for each layer. It is important that the correct answer be provided to this question, as array reading and writing conventions are different for these two data types.

Next MOD2ARRAY asks for the name of the MODFLOW or MT3D (or SEAWAT) name file associated with the current model. The prompt is:-

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

MOD2ARRAY then reads this file, establishing that the nominated MODFLOW input file is indeed cited in it. It also makes a list of unit numbers employed in the current MODFLOW/MT3D (or SEAWAT) dataset. These are listed in the second column of the name file.

MOD2ARRAY's next prompt is:-

```
Enter filename base for array output files:
```

Suppose that a filename base of "*base*" is supplied in response to the above prompt. MOD2ARRAY will write extracted real arrays to files named *base1.ref*, *base2.ref*, *base3.ref*…. *basen.ref*…etc where *n* is the layer number associated with the array. Integer arrays will be written to files *base1.inf, base2.inf, base3.inf*…. *basen.inf*…etc. Where an array is not associated with a layer, the name of the file in which the array is stored is simply *base.ref* or *base.inf*.

Recall from Part A of this manual that the extension "*ref*" stands for "real formatted" while the extension "*inf*" stands for "integer formatted". Unformatted file storage by MOD2ARRAY is not allowed.

If the COLROW setting in file *settings.fig* has been set to "yes" indicating the need for members of the Groundwater Data Utilities suite to write a number-of-columns, number-of-rows header to integer and real array files, MOD2ARRAY checks that this is indeed the user's wish in the present case by asking:-

```
Include NCOL/NROW header in these files? [y/n]:
```

If it is intended that MODFLOW or MT3D read these files, then the response to this prompt should definitively be "no". However if it is required that other utility programs read them, and these programs expect a number-of-columns, number-of-rows header, then an answer of "yes" is more appropriate.

As well as writing array files, MOD2ARRAY rewrites the MODFLOW/MT3D input file from which arrays are extracted, with headers for extracted arrays replaced by

headers that direct MODFLOW or MT3D's attention to pertinent external array files. It asks for the name of the new MODFLOW/MT3D input file using the following prompt:-

```
Enter name for altered MODFLOW/MT3D input file:
```

Two conventions are available for directing MODFLOW/MT3D's attention to external files in order to read arrays. The first is activated through using an OPEN/CLOSE statement in the array header, followed by the name of the file holding the array. The second is activated through provided a unit number different from that with which the MODFLOW input file is being read, and through citing that unit number together with a DATA specifier and the pertinent filename in the MODFLOW or MT3D (or SEAWAT) name file. The first option is only available for MODFLOW input files, whereas the second option is available for both MODFLOW and MT3D input files. The first is preferable, however, as it does not require that all array files be open from the moment of commencement of MODFLOW execution. Hence the response to MOD2ARRAY's next prompt, which is:

```
Use OPEN/CLOSE or DATA convention for array headers in this file  [o/d]:
```

should be "o" if the input file from which arrays are being extracted is a MODFLOW input file, and "d" if it is an MT3D input file. (It is hoped that the OPEN/CLOSE convention will one day become available in MT3D; hence the above question is specifically asked.)

MOD2ARRAY's final prompt is:-

```
Enter name for altered MODFLOW/MT3D name file:
```

In the new name file, the name of the new MODFLOW/MT3D input file replaces that of the old one. Also, if the "DATA" option is supplied in response to the above prompt, the names of array files and corresponding unit numbers are added to the name file.

Once it has received all of the above information, MOD2ARRAY goes about its business of extracting arrays, writing them to pertinent array files, re-writing the MODFLOW/MT3D input file, and altering the name file. Once this task is complete, MODFLOW, MT3D (or SEAWAT) can be immediately run on the basis of the new name file.

Uses of MOD2ARRAY

Arrays are the building blocks of MODFLOW and MT3D datasets. In most parameter estimation contexts, it is convenient for some of these arrays to be stored in dedicated files so that they can be written by model preprocessing software using current parameter values on each occasion that the model is run. Sometimes more complex processing can be carried out involving many such arrays (see for example the ELEV2CONC and PARM3D utilities). All model preprocessors and the model itself (which may be comprised of MODFLOW, MT3D, both of these together, SEAWAT, or some other program) are then run from a batch or script file. In estimating parameters for these model(s), PEST calls this batch file many times; hence its name

is provided to PEST through the “model command line” section of the PEST control file.

See Also

See also ELEVCONC, PARM3D, FAC2REAL, INT2REAL and TWOARRAY.

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 MOD2OBS_DBL, MOD2SMP, MOD2DAT, PESTPREP, PESTPREP1, PESTPREP2 and SMP2SMP.

MOD2OBS_DBL

Function of MOD2OBS_DBL

MOD2OBS_DBL performs an identical role to that of MOD2OBS. The only difference is that it reads a MODFLOW system state file that is recorded in double precision. Such a file is written by MODFLOW 6 . Hence MOD2OBS_DBL can be used as a postprocessor for a MODFLOW 6 model which employs a structured (i.e. DIS) grid.

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.

Using 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 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 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.

See Also

See also BUD2SMP, MOD2SMPDIFF, MOD2OBS, PESTPREP and SMP2SMP.

MOD2SMPDIFF

Function of MOD2SMPDIFF

The operation of MOD2SMPDIFF is similar to that of MOD2SMP in that it reads a MODFLOW or MT3D unformatted output file and undertakes spatial interpolation from the arrays recorded in that file to the sites of bores. Such interpolation is undertaken at every simulation time represented in the MODFLOW/MT3D output file. Outcomes or this interpolation process are recorded in bore sample file format. The difference between MOD2SMPDIFF and MOD2SMP is that the former program computes differences or ratios between MODFLOW/MT3D outputs at user-nominated sites. These differences or ratios are then recorded in site sample file format rather than the individual heads or concentrations read from the MODFLOW/MT3D output files.

Using MOD2SMPDIFF

MOD2SMPDIFF has the following in common with most programs of the Groundwater Data Utility suite.

1. A settings file named *settings.fig* must be present in the directory from which it is run. This must inform MOD2SMPDIFF of the date protocol which it must use in writing its output bore sample file ("dd/mm/yyyy" or "mm/dd/yyyy").

2. If the response to any prompt is "e" followed by the <Enter> key, MOD2SMPDIFF will backtrack to its previous prompt.

3. If a filenames file (named *files.fig*) is present in the current directory, MOD2SMPDIFF will look in that file for the name of the current bore coordinates file. (This is optional).

MOD2SMPDIFF's first two prompts are the same as those of MOD2SMP, namely:-

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

Then, instead of asking for the name of a bore listing file, it asks for the name of a "bore difference listing file". An example of such a file follows.

```
B12321        B12322      DIFF
BH35          BH23        DIFF35-23
BH23U         BH23L       BH23U-L
```

Part of a bore difference listing file.

Each line of a bore difference listing file must contain three entries. The first two are the identifiers pertaining to two bores; each of these must figure in the previously-named bore coordinates file. The third entry is a new name. This is the identifier

which will be associated with the difference or ratio of heads/concentrations interpolated from MODFLOW/MT3D arrays to the sites of the first two bores. As usual, a bore identifier must be 20 characters or less in length. If this is protocol is not observed, MOD2SMPDIFF will cease execution with an error message.

MOD2SMPDIFF next asks:-

```
Take ratio or difference of first to second column of this file?  [r/d]:
```

Answer with "r" or "d" as appropriate. In the first case MOD2SMPDIFF calculates, at each of its output times, the difference between the head/concentration at the first bore cited on each line of the bore difference listing file and that at the second respective bore cited on the same line. In the second case the ratio between the two (first over second) is computed.

Next MOD2SMPDIFF prompts for the name of the MODFLOW or MT3D unformatted output file which it must read:-

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

As stated above, differences or ratios for heads/concentrations are computed at every output time represented in this file. Because the array headers are different for MODFLOW and MT3D output files, MOD2SMPDIFF needs to know what type of file it is being asked to read. So it prompts:-

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

and then:-

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

Respond to this prompt with a number equal to or greater than the number of output times for which arrays are recorded in the file. MOD2SMPDIFF needs to know this so that it can dimension arrays prior to reading the file. If you are unsure of this number, use the ARRDET program to ascertain the contents of the file. Alternatively, enter a number that errs on the high side. If you err on the low side do not worry; MOD2SMPDIFF will inform you if there are arrays corresponding to more simulation times than you had anticipated.

MOD2SMPDIFF's next prompt is:-

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

Any head or concentration whose absolute value is above this number is assumed to represent an inactive or dry cell. MOD2SMPDIFF adjusts its interpolation mechanism to account for these.

The remainder of MOD2SMPDIFF's prompts are the same as those of MOD2SMP, namely:-

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

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

Enter name for bore sample output file:
```

After having received all the information that it requires from the user, MOD2SMPDIFF reads the nominated MODFLOW/MT3D output file and records its calculated differences or ratios to the nominated bore sample output file.

The following should be noted.

1. If either of the nominated bores comprising the first two identifiers on a particular line of the bore difference listing file lie within an inactive cell, dry cell, or outside of the grid altogether, the difference or ratio is assigned the value 1.1E37.

2. If a ratio is sought and the denominator (i.e. the head or concentration corresponding to the second bore on a line of the bore difference listing file) is 0.0, the ratio is provided as 1.0E30 or -1.0E30 depending on the sign of the numerator, unless the latter is zero, in which case the ratio is provided as 1.0.

Uses of MOD2SMPDIFF

There are occasions in model calibration where PEST should be asked to look at differences rather than (or in addition to) absolutes. For example, it is often advantageous to explicitly introduce the differences between heads in different layers to the parameter estimation process, assign these differences to their own observation group, and then ensure (through proper weights assignment) that the "visibility" of this group in the objective function is sufficient for PEST to take notice of it. This may save these all-important differences from being "drowned" in measurement noise, thereby allowing better estimates of the vertical conductance of an aquitard to be gained.

See Also

See also ARRDET, MOD2OBS, MOD2SMP, SMPDIFF and SMPTREND.

PARCOV

Function of PARCOV

PARCOV reads a file containing parameter names and geographical coordinates. It also reads a geostatistical structure file containing one or a number of structure and variogram specifications. It writes a covariance matrix for the parameters supplied in the first of these files on the basis of a geostatistical structure supplied in the second.

Using PARCOV

PARCOV commences execution by prompting for the name of a "parameter coordinates file":

```
Enter name of parameter coordinates file:
```

A parameter coordinates file should contain three columns of data. The first column should be comprised of parameter names. The second and third columns should be comprised of eastings and northings associated with respective parameters.

PARCOV next prompts:-

```
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 the parameters featured in the parameter coordinates file in response to the following prompt:-

```
Enter structure to use for parameters:
```

PARCOV's final prompt is:-

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

The name of a file to which the covariance matrix is written is supplied in response to this prompt. The format of this file is the same as that used by PEST matrix manipulation utilities. See PEST documentation for details.

Elements of the covariance matrix assigned to parameters cited in the parameter coordinates file do not depend on whether the geostatistical structure with which these parameters are associated is provided with a TRANSFORM status of "none" or "log". However consistency is important. Thus any PEST control file which cites these same parameters (and for which the PARCOV-generated covariance matrix provides a statistical characterization), must also cite these parameters as log-transformed or untransformed, in accordance with the settings of the geostatistical structure on which basis the covariance matrix was constructed.

Uses of PARCOV

PARCOV can be used for building a C(**p**) (i.e. a parameter covariance) matrix for the use of PEST utilities such as RESPROC, PREDERR, PARAMERR, members of the PREDVAR suite of utilities, and members of the PREDUNC utility suite. As such it furnishes the basis for calculation of the contribution to predictive error variance and/or uncertainty made by the inability of the model calibration process to capture system hydraulic property detail. In most modelling contexts this is the dominant contributor to model predictive error.

See Also

See also PPCOV, PPCOV3D and PPCOV_SVA.

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
```

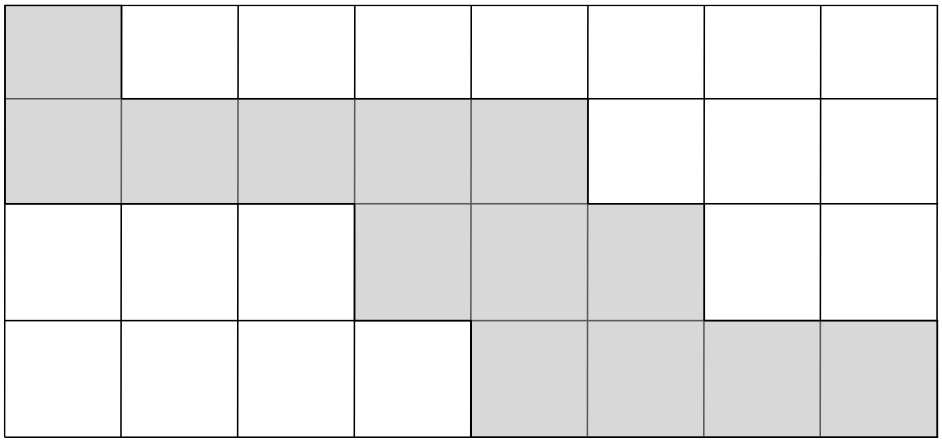
Examples of zone value assignment instructions.

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 17. 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 twenty 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 and SMP2SMP.

PESTPREP1

Function of PESTPREP1

The function of PESTPREP1 is identical to that of PESTPREP. That is, it builds a complete PEST input dataset based on an observed bore sample file and its model-generated equivalent, together with parameters cited in one or more template files. However it is built for use in conjunction with PESTPREP2, the function of the latter program being to add more observations to the PEST input dataset. These observations may be based on the same bores as those used by PESTPREP1, but reference different data types. PESTPREP1 therefore provides an observation naming convention that allows these different observations types to be distinguished from each other; it also allows the user to assign observations to an observation group name of his/her choosing instead of the default observation name used by PESTPREP.

Using PESTPREP1

Use of PESTPREP1 is identical to that of PESTPREP, except for a few minor changes. These are now outlined.

PESTPREP1 names observations in a slightly different manner to that in which PESTPREP names observations. For PESTPREP1 each observation name must begin with a user-supplied prefix of eight characters or less. The remainder of the name is formed in the same manner as for PESTPREP, i.e. using sequential numbers, or bore identifiers followed by the observation number associated with each bore. The PESTPREP1 naming convention is useful where the PEST input dataset includes more than one observation type associated with each bore (with these additional observations being added to the PESTPREP1-generated PEST input dataset by PESTPREP2), for the prefix then distinguishes one observation type from another.

If the naming-by-bore-identifier option is selected, PESTPREP1, like PESTPREP, asks:-

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

where *n* is calculated so that the observation name (including prefix and bore observation number) fits within the 20 character observation name limit imposed by PEST. It is possible that *n* in the above prompt will exceed the 20 character limit allowed for bore identifiers. If this is the case, the prompt is unnecessary of course. However it is retained in case, at some future date, the 20 character bore identifier limit is lifted.

The other feature that distinguishes PESTPREP1 from PESTPREP is the fact that PESTPREP1 assigns observations to an observation group whose name is specified by the user, rather than to the group "obsgroup" used by PESTPREP. This measure, too, facilitates the addition of other observation types by PESTPREP2; these other observation types should naturally be assigned to different observation groups.

Uses of PESTPREP1

Uses of PESTPREP1 are the same as those of PESTPREP. However a PESTPREP1-generated PEST control file is more amenable to the later addition of further observation data by PESTPREP2.

See Also

See also, PESTPREP, PESTPREP2, MOD2OBS.

PESTPREP2

Function of PESTPREP2

PESTPREP2 is designed to be used in conjunction with PESTPREP1. It allows extra observations to be added to an existing PEST input dataset, rather than the creation of a new PEST input dataset (which is the job of PESTPREP1). As for PESTPREP and PESTPREP1, observations are assumed to reside in a bore sample file; model-generated equivalents to observations (produced by MOD2OBS or SMP2SMP) are also assumed to reside in a bore sample file.

Using PESTPREP2

Use of PESTPREP2 is almost identical to that of PESTPREP1 (and hence to PESTPREP). A typical set of prompts and responses is shown below.

```
Enter name of observation bore sample file: observ.smp
Enter name of    model    bore sample file: model.smp

Enter prefix for new observation names (8 chars or less): conc_
Use numbers or bore identifiers for rest of observation names?  [n/b]: b
Use first 13 or last 13 characters of bore identifier?  [f/l]: f
Enter name for new observation group: concs

Enter name for instruction file: concs.ins
- file concs.ins written ok.

Enter name of existing PEST control file: case1.pst
Enter name for new PEST control file: case2.pst
- file case1.pst read ok.
- file case2.pst written ok.
```

Prompts issued by PESTPREP2 depart from those issued by PESTPREP1 where the former program asks for the name of an existing PEST control file. In contrast, at this stage of its operations PESTPREP1 prompts for the names of template files from which it reads parameter names. PESTPREP2 does not add any parameters to an existing PEST input dataset. However it adds all observations formulated from the model bore sample file which it reads (the "measured values" of which reside in the complimentary observation bore sample file). It also adds the new instruction file, together with the name of the model output file which these instructions are designed to read (i.e. the "model bore sample file" whose name is requested in the second of the above prompts) to the PEST input dataset, as well as the observation group to which these new observations are assigned. All other components of the existing PEST input dataset are left unchanged. (It is thus the user's responsibility to add any new commands, such as that required to run MOD2OBS or SMP2SMP, to the model batch file.)

It goes without saying that the PEST input dataset produced as an outcome of PESTPREP2 execution should be checked using the PESTCHEK utility before PEST is run.

Like PESTPREP and PESTPREP1, PESTPREP2 assigns a weight of 1 to all new observations. The ADJOBS utility can be employed to modify these weights, this being especially useful if it is desired that weights be a function of observed data value. PEST utilities such as the PWTADJ1 utility can also be of use in the assignment of differential weights to the various observation groups involved in the parameter estimation process.

Uses of PESTPREP2

As stated above, PESTPREP2 is a partner to PESTPREP1. Collectively they can be used to create a complex PEST input dataset comprised of many different observation types.

See Also

See also PESTPREP, PESTPREP1, MOD2OBS 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 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 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 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 20 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

PPCOV3D can be used for building a C(**p**) (i.e. a parameter covariance) matrix for the use of PEST utilities such as those belonging to the PREDVAR and PREDUNC suite of programs. As such, it furnishes the basis for calculation of the contribution to predictive error variance and/or predictive uncertainty 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/uncertainty.

The covariance matrix written by PPCOV3D can also be used by PEST itself. It can be ascribed to an observation group (including a prior information group) used in Tikhonov regularization. Thus if heterogeneity is introduced to the model domain through the inversion process, its introduction is in accordance with the underlying parameter variogram(s) inasmuch as this is not violated by measurements used in the calibration process.

See Also

See also PPCOV_SVA, PARCOV, PPCOV3D, PPCOV_SVA, PPK2FAC, FIELDGEN and GENREG.

PPCOV_SVA

Function of PPCOV_SVA

PPCOV_SVA ("SVA" stands for "spatially varying anisotropy") performs a similar role to that of the PPCOV utility in that it builds a covariance matrix for parameters which are associated with a set of two-dimensional pilot points. However it does not rely on an assumption of geostatistical stationarity. In fact, variogram properties can be different at the location of every pilot point. Construction of the covariance matrix in such a context of spatially varying geostatistical properties can only be approximate. Nevertheless the covariance matrix produced by PPCOV_SVA may prove useful in implementing regularisation, and in generation of random values for pilot point parameters where pilot points are used to represent spatial hydraulic property heterogeneity in a model domain of highly variable geology.

Using PPCOV_SVA

Unlike PPCOV and PPCOV3D, PPCOV_SVA does not read a structure file in order to obtain the properties of a geostatistical structure that is assumed to characterize hydraulic property heterogeneity throughout a model domain. Recall that a geostatistical structure can include one or a number of variograms; these are then nested to characterize total spatial variability. Instead, a single variogram type is assumed to prevail throughout the model domain. However its sill, nugget, range, anisotropy and anisotropy direction can vary on a pilot-point-by-pilot-point basis. Pilot point specific nuggets can also be employed.

PPCOV_SVA reads a "pilot points statistical specification" file. This resembles a normal pilot points file. However it contains extra data columns; it can also optionally contain a header line at its top. An example of the first part of a pilot points statistical specification file is shown below.

point	x	y	zone	nugget	sill	a	hanis	bearing
ppt1	35.0	765.0	1	0.5	0.8	350.0	4.0	45
ppt2	95.0	765.0	1	0.5	0.8	350.0	4.0	45
ppt3	155.0	765.0	1	0.5	0.8	350.0	4.0	45
ppt4	215.0	765.0	1	0.5	0.8	350.0	4.0	45
ppt5	275.0	765.0	1	0.5	0.8	350.0	4.0	90
ppt6	335.0	765.0	1	0.0	0.8	350.0	4.0	90
ppt7	395.0	765.0	1	0.0	0.8	350.0	4.0	90
ppt8	455.0	765.0	1	0.0	0.5	350.0	4.0	90
ppt9	35.0	705.0	1	0.0	0.5	150.0	0.0	0
ppt10	95.0	705.0	1	0.0	0.5	150.0	0.0	0
ppt11	155.0	705.0	1	0.0	0.5	150.0	0.0	0
ppt12	215.0	705.0	1	0.0	0.3	150.0	0.0	0
ppt13	275.0	705.0	2	0.0	0.3	150.0	0.0	0
ppt14	335.0	705.0	2	0.0	0.3	150.0	0.0	0
ppt15	395.0	705.0	2	0.0	0.3	150.0	0.0	0
etc								

The first part of a pilot points statistical specification file.

Column headers can be provided as the first line of the pilot points statistical specification file. This line is optional; if it supplied, PPCOV_SVA does not read it. It exists purely for the benefit of the user.

The first four columns of a pilot points statistical specification file are the same as those of a normal pilot points file. They contain, respectively, pilot point identifiers (of 12 characters or less), pilot point eastings and northings, and pilot point zone numbers. The ensuing columns must contain variogram specifications; the correct order of these columns is shown in the above example. First comes the nugget; then come the variogram sill, the "a" value of the variogram (this is proportional to its range), and then the horizontal anisotropy and anisotropy bearing (with respect to north) of the variogram. See descriptions of the geostatistical structure file provided elsewhere in this documentation for a full explanation of these variables. Note that if anisotropy is greater than 1.0, then the variogram "a" value describes the range of the variogram in the direction of greatest variogram elongation; meanwhile the bearing of anisotropy points in this same direction.

If pilot point parameters are log-transformed, then all of the variogram characteristics specified in the pilot points statistical specification file must pertain to the log (to base 10) of parameter values. PPCOV_SVA has no knowledge of the transformation status of pilot point parameters; hence it does not check this.

Use of PPCOV_SVA is very similar to that of PPCOV, except for the requirement that a pilot points statistical specification file be provided instead of a pilot points file and a geostatistical structure file. Prompts and typical PPCOV_SVA responses are as follows.

```
Enter name of pilot points statistical specs file: hk_stat.pts
Skip a line a the top of this file?  [y/n]: y
 - data for 103 pilot points read from pilot points file hk_stat.pts

Enter minimum allowable separation for points in same zone: 0.0

Is overall variogram spherical, exponential or Gaussian? [s/x/g]: s

Enter name for output matrix file: cov.dat
Enter pilot point prefix for parameter name (<Enter> if none): k_

Filling covariance matrix....
Using SVD to assure positive definiteness of matrix....
- file temp.dat written ok.
```

The algorithmic basis of PPCOV_SVA is very simple. The total variance assigned to any pilot point is the sum of the nugget and the sill that are ascribed to that pilot point. If two pilot points lie in separate zones, then their covariance is zero. If they lie in the same zone, a covariance between them is calculated using the variogram characteristics pertaining to both pilot points; the lesser of these covariances is then adopted. Positive definiteness of the resulting matrix is then guaranteed by subjecting it to singular value decomposition and rebuilding it after equating the **V** matrix to the **U** matrix.

Uses of PPCOV_SVA

The covariance matrix produced by PPCOV_SVA can be used in conjunction with the PEST ADDREG1 utility to implement preferred value regularisation in areas of complex geology. It can also be used with the PEST RANDPAR utility, and with PLPROC, to generate random values for pilot point parameters.

See Also

See also PPCOV, PPCOV3D, PPCOV_SVA, PARCOV and MKPPSTAT.

PPCOV3D

Function of PPCOV3D

PPCOV3D reads a three-dimensional 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 PPCOV3D

PPCOV3D 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 pilot points file must be of the three-dimensional type. The first column of this file must contain pilot point names (10 characters or less in length). The next three columns must contain pilot point eastings, northings and elevations. Following these is a column of zone numbers, followed by another column containing the values assigned to pilot points. PPCOV3D ignores data provided in this last column. 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.

PPCOV3D'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 PPCOV3D:-

```
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. Each structure that is named through responses to the above prompts must be of the three-dimensional type. Thus the variograms which it employs must cite the *ang1*, *ang2*, *ang3*, *a_hmax*, *a_hmin* and *a_vert* parameters which characterize variograms of this type.

PPCOV3D'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 PPCOV3D

PPCOV3D can be used for building a C(**p**) (i.e. a parameter covariance) matrix for the use of PEST utilities such as those belonging to the PREDVAR and PREDUNC suite of programs. As such, it furnishes the basis for calculation of the contribution to predictive error variance and/or predictive uncertainty 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/uncertainty.

The covariance matrix written by PPCOV3D can also be used by PEST itself. It can be ascribed to an observation group (including a prior information group) used in Tikhonov regularization. Thus if heterogeneity is introduced to the model domain through the inversion process, its introduction is in accordance with the underlying parameter variogram(s) inasmuch as this is not violated by measurements used in the calibration process.

See Also

See also PARCOV, PPCOV, PPCOV_SVA, PPK2FAC, FIELDGEN and GENREG.

PPCOV3D_SVA

Function of PPCOV3D_SVA

PPCOV3D_SVA ("SVA" stands for "spatially varying anisotropy") performs a similar role to that of the PPCOV3D utility in that it builds a covariance matrix for parameters which are associated with a set of three-dimensional pilot points. However it does not rely on an assumption of geostatistical stationarity. In fact, variogram properties can be different at the location of every pilot point. Construction of the covariance matrix in such a context of spatially varying geostatistical properties can only be approximate. Nevertheless the covariance matrix produced by PPCOV3D_SVA may prove useful in implementing regularisation, and in generation of random values for pilot point parameters where pilot points are used to represent spatial hydraulic property heterogeneity in a model domain of highly variable geology.

Using PPCOV3D_SVA

Unlike PPCOV and PPCOV3D, but similarly to PPCOV_SVA, PPCOV3D_SVA does not read a structure file in order to obtain the properties of a geostatistical structure that is assumed to characterize hydraulic property heterogeneity throughout a model domain. Recall that a geostatistical structure can include one or a number of variograms; these are then nested to characterize total spatial variability. Instead, a single variogram type is assumed to prevail throughout the model domain. However its sill, nugget, three-dimensional ranges and three-dimensional anisotropy directions can vary on a pilot-point-by-pilot-point basis. Pilot point specific nuggets can also be employed.

PPCOV3D_SVA reads a "three-dimensional pilot points statistical specification" file. This resembles a normal pilot points file. However it contains extra data columns; it can also optionally contain a header line at its top. An example of the first part of a three-dimensional pilot points statistical specification file is shown below.

point_id	x	y	z	zone	nugget	sill	a_hmax	a_hmin	a_vert	ang1	ang2	ang3
1_1	35.0	765.0	10.0	1	0.0	0.5	100.0	30.0	10.0	20.0	0.0	0.0
1_2	95.0	765.0	10.0	1	0.0	0.5	100.0	30.0	20.0	20.0	0.0	0.0
1_3	155.0	765.0	10.0	1	0.0	0.5	100.0	30.0	10.0	30.0	10.0	0.0
1_4	215.0	765.0	20.0	1	0.0	0.5	30.0	30.0	10.0	30.0	10.0	0.0
1_5	275.0	765.0	20.0	1	0.0	0.5	30.0	10.0	20.0	80.0	10.0	0.0
1_6	335.0	765.0	25.0	1	0.0	0.5	30.0	10.0	10.0	80.0	0.0	0.0
1_7	395.0	765.0	25.0	2	0.0	0.25	30.0	10.0	10.0	135.0	20.0	0.0
1_8	455.0	765.0	30.0	2	0.0	0.25	200.0	10.0	10.0	135.0	20.0	0.0
1_9	35.0	705.0	30.0	2	0.0	0.25	200.0	30.0	10.0	135.0	20.0	0.0
1_10	95.0	705.0	30.0	2	0.0	0.25	200.0	30.0	20.0	90.0	20.0	0.0
1_11	155.0	705.0	40.0	2	0.0	0.25	200.0	30.0	20.0	90.0	20.0	0.0
etc												

The first part of a three-dimensional pilot points statistical specification file.

Column headers can be provided as the first line of the three-dimensional pilot points statistical specification file. This line is optional; if it is supplied, PPCOV3D_SVA does not read it. It exists purely for the benefit of the user.

The first five columns of a three-dimensional pilot points statistical specification file are the same as those of a normal three-dimesional pilot points file. They contain, respectively, pilot point identifiers (of 12 characters or less), pilot point eastings, northings, elevations, and pilot point zone numbers. The ensuing columns must contain three-dimesional variogram specifications; the correct order of these columns is shown in the above example. First comes the nugget; then comes the variogram sill. Columns containing the variogram *a_hmax*, *a_hmin* and *a_vert* values follow this; these are the "a" values of the three-dimensional variogram in two orthogonal, roughly horizontal, directions and in a third, roughly vertical, direction. The angles which specify these directions (namely *ang1*, *ang2* and *ang3*) follow. Refer to the documentation of the PPKFAC3D utility for a description of all of these three-dimensional variogram specifications.

If pilot point parameters are log-transformed, then all of the variogram characteristics specified in the three-dimensional pilot points statistical specification file must pertain to the log (to base 10) of parameter values. PPCOV3D_SVA has no knowledge of the transformation status of pilot point parameters; hence it does not check this.

Use of PPCOV3D_SVA is very similar to that of PPCOV3D, except for the requirement that a three-dimensional pilot points statistical specification file be provided instead of a pilot points file and a geostatistical structure file. Prompts and typical PPCOV3D_SVA responses are as follows.

```
Enter name of 3D pilot points statistical specs file: pp_specs.dat
Skip a line at the top of this file?  [y/n]: y
 - data for 312 pilot points read from file pp_specs.dat

Is overall variogram spherical, exponential or Gaussian? [s/x/g]: x

Enter name for output matrix file: temp.dat
Enter pilot point prefix for parameter names (<Enter> if none): k_

Filling covariance matrix....
Using SVD to assure positive definiteness of matrix....
- file temp.dat written ok.
```

The algorithmic basis of PPCOV3D_SVA is very simple. The total variance assigned to any pilot point is the sum of the nugget and the sill that are ascribed to that pilot point. If two pilot points lie in separate zones, then their covariance is zero. If they lie in the same zone, a covariance between them is calculated using the variogram characteristics pertaining to both pilot points; the lesser of these covariances is then adopted. Positive definiteness of the resulting matrix is then guaranteed by subjecting it to singular value decomposition and rebuilding it after equating the **V** matrix to the **U** matrix.

Uses of PPCOV3D_SVA

The covariance matrix produced by PPCOV3D_SVA can be used in conjunction with the PEST ADDREG1 utility to implement preferred value regularisation in areas of complex geology. It can also be used with the PEST RANDPAR utility, and with PLPROC, to generate random values for pilot point parameters.

See Also

See also PPCOV, PPCOV3D, PPCOV_SVA, PARCOV and MKPPSTAT3D.

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. 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.

Using 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
END VARIOGRAM
```

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”.

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 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 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 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 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 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 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 such as FAC2REAL 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 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 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. However the pilot points file supplied to this program 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 FAC2REAL. FAC2REAL 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.

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 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.

Using 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.

PPK2FACG

Function of PPK2FACG

PPK2FACG is similar to other members of the PPK2FAC* family in that it generates a set of kriging factors through which interpolation can take place from a set of pilot points to another set of user-nominated points. Its complimentary program FAC2G then undertakes this interpolation based on these factors. "G" stands for "general", as the format for the file containing the points to which interpolation must take place is not related to that of any specific model. However in most cases these will represent the locations of the nodes of a numerical grid or mesh.

PPK2FACG also writes regularization information to a user-nominated file. This can be used by the PPKREG utility for adding preferred-difference prior information equations to an existing PEST control file. The parameters that are cited in these prior information equations are assumed to be associated with the set of pilot points from which interpolation takes place.

Using PPK2FACG

Prompts, and typical responses to PPK2FACG prompts, are as follows.

```
Enter name of pilot points file: vk5.pts
 - data for 193 pilot points read from pilot points file vk5.pts
Enter minimum allowable points separation: 0

Enter name of nodal x,y,zone file: laymat5.xyz

Enter name of structure file: struct.dat

The following zones have been detected in x,y,zone file:-

   For zone characterised by integer value of 5:-
   Enter structure name (blank if no interpolation for this zone): struc_1
   Perform simple or ordinary kriging [s/o]: o
   Enter search radius: 1e20
   Enter minimum number of pilot points to use for interpolation: 1
   Enter maximum number of pilot points to use for interpolation: 30

   For zone characterised by integer value of 6:-
   Enter structure name (blank if no interpolation for this zone):<Enter>

Enter name for interpolation factor file: factors.dat
Is this a formatted or unformatted file? [f/u]: f

Enter name for regularisation information file: reg.dat
```

The format of an "x,y,zone" file is illustrated by the following example.

```
722855.0930     4447533.306      10
723081.6700     4447427.652      10
723534.8240     4447216.342      5
723761.4000     4447110.688      5
723987.9780     4447005.034      5
724214.5550     4446899.378      5
724441.1320     4446793.724      5
722976.0160     4447201.075      9
723429.1690     4446989.765      5
723655.7460     4446884.111      5
723882.3230     4446778.456      5
724108.9000     4446672.802      5
724335.4770     4446567.147      5
724562.0540     4446461.493      5
726827.8230     4445404.947      5
727054.3990     4445299.292      5
727280.9780     4445193.638      5
727507.5540     4445087.983      5
727734.1320     4444982.328      5
722190.6300     4447291.461      8
```

Part of an x,y,zone file.

As is apparent from the above figure, an x,y,zone file contains three columns of data. The first two columns contain the eastings and northings of points to which interpolation must take place. The third column lists the zone number that is associated with each point.

Interpolation factors written by PPK2FACG are readable by the FAC2G utility. This uses the kriging factors computed by PPK2FACG to interpolate from pilot points to the points cited in the x,y,zone file.

All other aspects of PPK2FACG use are identical to that of PPK2FAC. Note in particular that interpolation to a point cited in the x,y,zone file will take place only from pilot points that belong to the same zone as that to which the point belongs.

See Also

See also FAC2G, 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.

PPK2FAC2

Function of PPK2FACF2

PPK2FAC2 is identical to PPK2FAC1 except for the fact that it prompts for a "blanking radius" in addition to an interpolation radius. If any cell for which interpolation is requested is removed from its closest pilot point by a distance that is greater than this distance, then the cell is "blanked", irrespective of the interpolation radius. Thus the latter may be very large in order to avoid discontinuities in the kriged field. However the blanking radius may be relatively small, in order to prevent computation of unrealistic interpolated values.

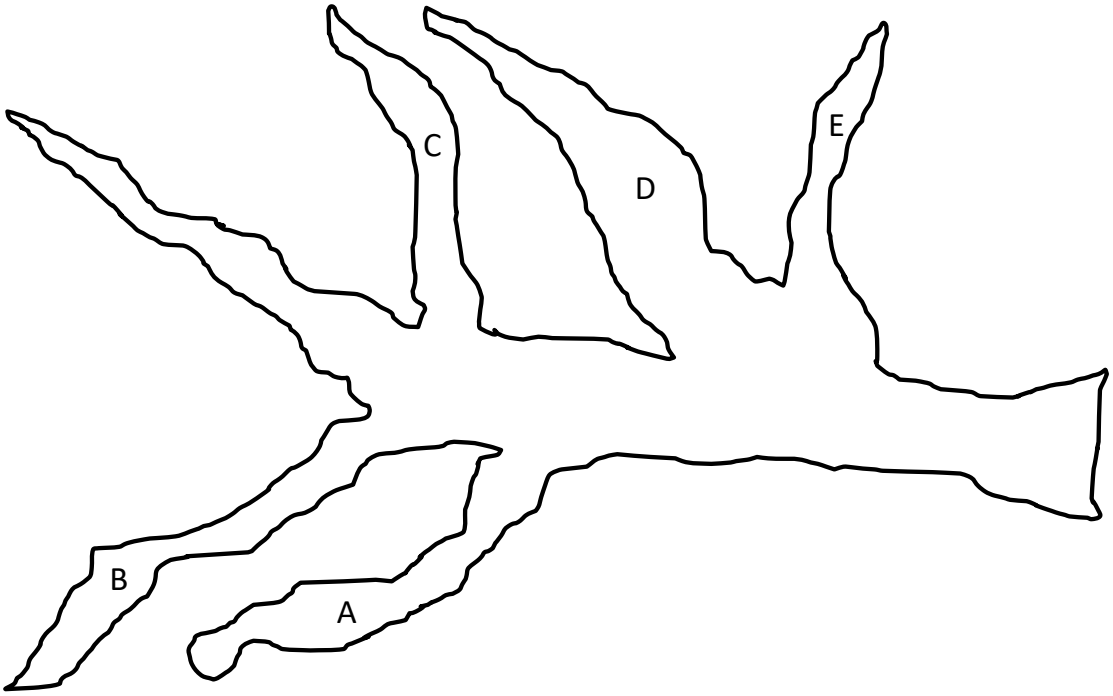
PPK2FAC2 does not calculate kriging factors for a blanked cell. When using FAC2REAL, this cell can then be assigned an "interpolated value" which is the same as that assigned to any other cell to which interpolation cannot take place (for example an indicator value such as 1.0e35).

PPK2FAC3

Function of PPK2FAC3

Like PPK2FAC, PPK2FAC1 and PPK2FAC2, PPK2FAC3 was designed for use in the MODFLOW/MT3D/SEAWAT modelling environments. However some features have been included in the design of PPK2FAC3 which may yield better interpolated hydraulic property fields in narrow alluvial systems, particularly those which are comprised of a main trunk and multiple tributaries.

Ideally in systems such as this, spatial interpolation should take place within tributaries, but should not cross from one tributary to another where the two are close but separate, as is illustrated in the figure below.



Model domain covering an alluvial valley network.

Pilot points may be placed throughout a model domain such as that shown above to allow expression of alluvial hydraulic property heterogeneity within the calibration process. In particular, pilot points may be placed down tributaries A and B (as well as through all other parts of the model domain). However interpolation should be such that grid cells within tributary A are informed only by pilot points which lie within this tributary (or in the main channel close to the mouth of the tributary), while grid cells in tributary B should be informed only by pilot points that also lie within that tributary (and perhaps by pilot points within the main channel that lie near the mouth of the tributary).

Using functionality available through PPKFAC, PPKFAC1 and PPKFAC2, this can be achieved through the use of zones. Thus tributaries A and B would be assigned to different zones. This mechanism would indeed ensure that no cross-tributary interpolation takes place. However it would also create discontinuities in hydraulic properties at zone boundaries - discontinuities which are unlikely to be present in the real alluvium system, and which a modeller may be trying to avoid through use of pilot points as a parameterization device.

PPK2FAC3 provides two mechanisms which can assist the modeller in overcoming this problem. Neither mechanism is perfect; so the user should monitor the performance of PPK2FAC3 carefully in assessing the effectiveness (or otherwise) of these mechanism in any particular modelling context. These mechanisms can be used together, or separately.

Like PPK2FAC, PPK2FAC1 and PPK2FAC2, PPK2FAC3 calculates and records factors on which kriging is ultimately based. Spatial interpolation is actually carried out by the FAC2REAL utility. Use of FAC2REAL in conjunction with PPK2FAC3-generated kriging factors is identical to its use with the other members of the PPK2FAC suite. Because PPK2FAC3 is built from PPK2FAC2 (which was built from PPK2FAC1), the PPKREG1 utility must be employed for adding "preferred parameter difference" regularization to a PEST control file, rather than the PPKREG utility, if this type of regularization is required. Alternatively, the all-purpose GENREG utility can be employed, or the PEST ADDREG1 utility in its stead if only "preferred parameter value" regularization is required.

Using PPK2FAC3

Exclusion Zones

Immediately after prompting the user for geostatistical structures and other variables governing interpolation within each of the zones that it finds in the integer array file with which it was provided (see documentation of PPK2FAC for details), PPK2FAC3 prompts:

```
Enter name of exclusion zone file (<Enter> if none):
```

If he/she wishes, the user can provide the name of an "exclusion zone file" in response to the above prompt. To prevent use of exclusion zone functionality, respond to the above prompt by simply pressing the <Enter> key.

An exclusion zone file is depicted in the figure below.

```
# This is an exclusion file
1    2  3  4  5
2    3  4  5  3  5 ! Sandy Creek Tributary
7    1  3  5
etc
```

Example of an exclusion zone file.

Any line in an exclusion zone file that begins with the "#" character is ignored Any characters including and following the "!" character on any line are ignored. Blank lines are also ignored.

All other characters in an exclusion file must be integers. Furthermore, all of these integers must feature in the integer zonation file that was previously read by PPK2FAC3. The first integer on each line denotes a "target zone"; the following integers denote "exclusion zones" pertinent to that target zone. Each line must possess at least two integers, the first identifying a target zone and the second identifying an exclusion zone. The target zone must have been assigned a geostatistical structure in response to previous PPK2FAC3 prompts; thus it must be a zone in which interpolation from pilot points to the finite-difference grid has been denoted as taking place.

Consider the first data line in the above exclusion zone file. The first integer is 1. The integers 2, 3, 4 and 5 follow. This line instructs PPK2FAC3 to calculate kriging factors based on the premise that cells in zone 1 of the finite-difference grid (this being the target zone) are assigned values through interpolating from all pilot points listed in the pilot points file except for the following:

1. pilot points which lie in zones to which no geostatistical structure was assigned in response to previous PPK2FAC3 prompts;

2. pilot points assigned to zones 2, 3, 4 and 5 (these being the exclusion zones pertinent to zone 1).

In the figure of the alluvial valley provided above, pilot points belonging to the zone ascribed to tributary B should be excluded from interpolation to cells within tributary A, even though some of the pilot points in tributary B may be closer to some of the cells in tributary A than some of the pilot points that have been assigned to this same tributary. However the zone comprising the main alluvial aquifer should not be excluded from interpolation to tributary A. Thus cells in the boundary area where tributary A meets the main aquifer will be informed by pilot points in both of these; hence the transition from the tributary to the main aquifer will be smooth.

The following should be noted.

1. Any target zone cited in an exclusion zone file must have been assigned a geostatistical structure in response to previous PPK2FAC3 prompts, this indicating that interpolation must take place to cells within that zone.

2. A target zone must not be cited as an exclusion zone for itself.

3. An exclusion zone cannot be assigned twice to the same target zone.

4. All zones which are NOT excluded from a particular target zone must have been assigned the same geostatistical structure as the target zone, except if they have been assigned no geostatistical structure at all. In the latter case interpolation will not take place from pilot points within pertinent zones to any

target zone, irrespective of the fact that they have not been specifically excluded from any target zone.

5. Not all target zones need to be listed in an exclusion zone file. In this case, pilot points from NO zones are excluded from interpolation to uncited target zones (except zones that have not been assigned a geostatistical structure as discussed above).

6. If all zones within a model domain are cited as target zones within an exclusion zone file, and if all zones but the target zone are listed as exclusion zones in each case, this provides identical functionality to that which would prevail if no exclusion zone file was supplied at all. In this case (as in normal PPK2FAC operation), interpolation to cells within a zone takes place only from pilot points within that zone.

7. If no exclusion zone file is supplied, PPK2FAC3 operates in an identical manner to PPK2FAC, PPK2FAC1 and PPK2FAC2. Thus interpolation to a cell is allowed only from pilot points that are assigned to the same zone as that in which the cell lines. However if an exclusion zone file is supplied (even if it cites only one target zone), then pilot points from ALL zones will be used for interpolation to all cells within any other zone, except for pilot points in zones that are specifically excluded from interpolation to user-specified target zones. Thus the whole *modus operandi* of zone functionality changes if an exclusion zone file is provided. Note also that, in light of the above comment, zones to which a certain geostatistical structure is assigned must be excluded from interpolation to a target zone to which a different structure is assigned, or an error condition will arise.

8. Irrespective of how many non-excluded zones exist for a particular target zone, interpolation to cells within the target zone will only take place from as many pilot points as the maximum allowed for that zone as provided by the user in response to previous PPK2FAC3 prompts. Likewise, the previously-provided interpolation radius (which is specific to each target zone) is still operative; no interpolation will take place from a pilot point to a cell if the separation between the two is greater than this distance. (Thus there is no need to exclude zones from a target zone if they are removed from the target zone by a distance that is greater than the interpolation radius associated with that zone.)

If an error condition is encountered, PPK2FAC3 will respond with an appropriate error message.

Interpolation Across Inactive Areas

Following PPK2FAC3's prompt for an exclusion file, it next prompts:

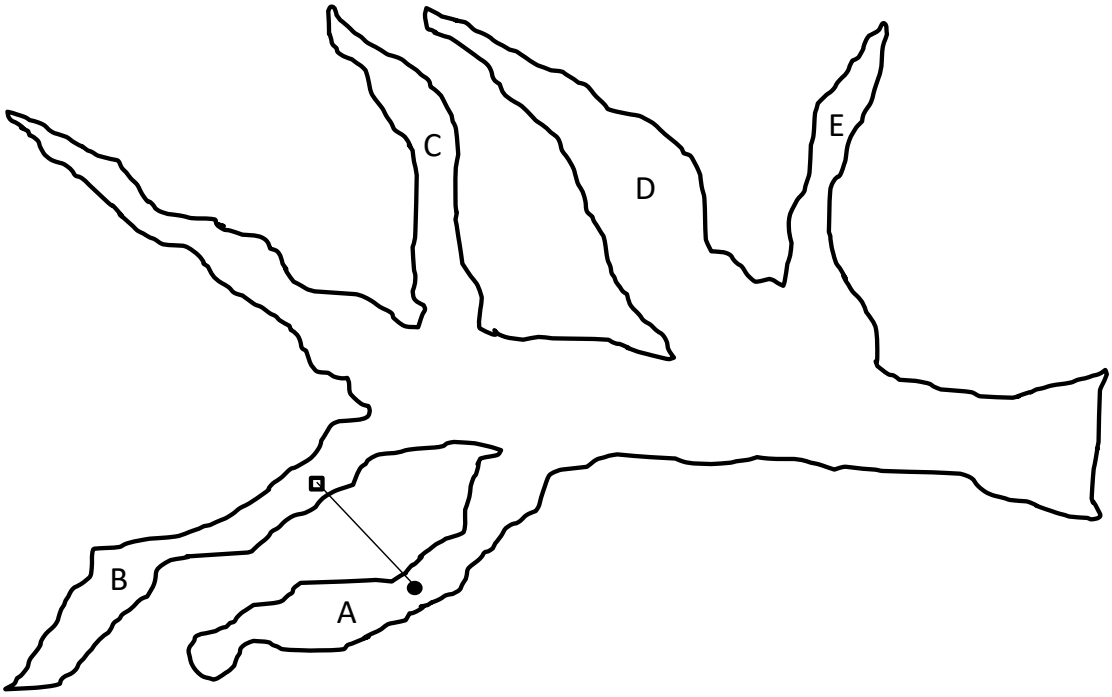
```
Prevent interpolation across inactive areas? [y/n]:
```

If the response to this question is "n", the matter is not pursued. However if it is "y", the following questions are asked.

```
Enter min relative contiguous inactive separation for rejection (0 to 1):
Enter min absolute contiguous inactive separation for rejection:
Reject if one or both thresholds is exceeded?  [o/b]:
```

On the basis of the user's answers to these questions, PPK2FAC3 decides what pilot points to exclude from interpolation to a particular cell. Note that this decision takes place in the context of whether or not exclusion zone functionality is operative. Thus pilot points can be excised from interpolation to a particular cell through this mechanism, only if they have not already been excised through zone considerations discussed above. For ease of characterization, this second exclusion methodology will be referred to as "line-of-sight exclusion".

Consider the pilot point and cell illustrated in the figure below. Suppose that both of these are in the same zone, or that they are in different zones but that the zone in which the pilot point lies has not been excluded from interpolation to the target zone in which the depicted cell lies.



A pilot point in tributary A and a grid cell in tributary B.

The line that joins the pilot point to the cell is also illustrated in the figure. Inclusion or exclusion of the pilot point as an information source for the depicted cell can take place according to the characteristics of this line if the response to the prompt:

```
Prevent interpolation across inactive areas? [y/n]:
```

is "y".

Suppose the user's response to the prompt:

```
Enter min relative contiguous inactive separation for rejection (0 to 1):
```

is then "0.1". Then the pilot point will not be an information source for the cell if there is a non-interrupted segment of the above line which is at least 0.1 of its length

and which lies within one or more zones that are excluded from informing the target zone to which tributary B belongs (these being denoted as "inactive areas" in the first of the above prompts). These "inactive" zone(s) may simply be those denoted in the previously-supplied integer array to which no geostatistical structure has been assigned; for example a zone characterized by an integer value of 0 is often used to indicate cells which are inactive from the model's point of view and hence require no hydraulic property assignment. Alternatively, the "inactive" zone from the line-of-sight exclusion point of view can be a zone that has been excluded from interpolation to zone B (if an exclusion zone file has been supplied) or is a zone other than that to which both the pilot point and the cell belong (if no exclusion zone file has been supplied).

Suppose the user's response to the prompt:

```
Enter min absolute contiguous inactive separation for rejection:
```

is "3560". Then the pilot point will be excluded from informing the cell if any uninterrupted segment of the line joining the two lies in an inactive zone (i.e. excluded from the target zone or without assignment of geostatistical structure) and is of length 3560 or more.

Through the last of the above three prompts, namely:

```
Reject if one or both thresholds is exceeded?  [o/b]:
```

the user can choose whether just one or both of the above conditions must be met before exclusion occurs.

The following should be noted.

1. Normal exclusion rules are applied before line-of-sight exclusion. Hence if a pilot point and a cell are separated by more than the user-supplied interpolation distance for the zone to which the cell belongs, or if there are *N* pilot points closer to the cell than the pilot point in question, where *N* is the user-supplied maximum number of pilot points used in interpolation within the target zone, exclusion will not take place according to line-of sight exclusion rules, for it will already have taken place in accordance with these other rules.

2. Line-of-sight exclusion may not always work as a user anticipates, especially where zones assigned to tributaries are thin and possess torturous boundaries. In this case a line joining a pilot point to a cell may cross inactive areas, even if the pilot point and the cell lie within the same tributary. In some cases this problem can be somewhat mitigated by defining the zone that defines interpolation to be somewhat wider than that which defines active tributary cells as used by the model. The fact that interpolation may then take place to cells for which hydraulic property values are not actually needed by the model is of no concern to the model.

3. For large model domains comprised of many cells, PPK2FAC3 may take a long time to run.

General

Irrespective of which of the above exclusion methodologies is employed, the user should check interpolation results carefully. The FAC2REAL utility (which undertakes spatial interpolation on the basis of kriging factors calculated by PPK2FAC3) asks the user for a value to assign to cells to which no interpolation takes place. The user should specify an obvious value (for example 1.0e35) as an indicator of this condition. He/she should then carefully inspect the real array that emerges from the interpolation process to ensure that cells in surprising places are not assigned this value. He/she should also check for discontinuities and/or flat areas in the interpolated field where all but one pilot point may have actually been excluded from interpolation.

The success of using the above exclusion methodologies will depend to a large degree on values assigned to other interpolation parameters (as will the success of the interpolation process in general). At the time of writing the following are recommended.

1. Choose an exponential variogram for interpolation. This is less likely to cause oscillatory behaviour of the interpolated field than other variograms.

2. Choose an "*a*" value for this variogram that is roughly equal to the average inter-pilot-point distance.

3. Do not leave large gaps in pilot point emplacement.

4. Provide an extremely high search radius.

5. However limit interpolation to the closest 9 to 15 pilot points.

6. Respect any pronounced anisotropy that may exists in an area.

Uses of PPK2FAC3

PPK2FAC3 contains all of the functionality that is provided in PPK2FAC, PPK2FAC1 and PPK2FAC2. However, as stated above, additional functionality available in PPK2FAC3 is designed specifically to overcome problems associated with long alluvial valleys.

At the time of writing PPK2FAC3 has had only limited application. User feedback is welcome.

See Also

See also FAC2REAL, PPKREG1.

PPK2FAC3D

Function of PPK2FAC3D

PPK2FAC3D computes kriging factors through which spatial interpolation can be undertaken from a set of three-dimensional pilot points to the cell-centres of a MODFLOW finite-difference grid. In this capacity it is very similar to the PPK2FAC family of utilities used for two-dimensional interpolation. However, unlike the latter utilities, it does not compute the interpolation standard deviation at each interpolated point. Nor does it store regularisation information for later use by utilities such as PPKREG which add regularisation prior information to a PEST control file.

PPK2FAC3D uses a modified version of the *kt3d* subroutine supplied with the GSLIB library in computation of kriging factors. See the following web page for more details.

http://www.gslib.com/

See also Deutsch and Journel (1987).

Using PPK2FAC3D

A Three-Dimensional Pilot Points File

The role of PPK2FAC3D is to assist in three dimensional interpolation from a set of three-dimensional pilot points to the nodes of a three-dimensional grid. Each pilot point must be supplied with a name (of 10 characters or less in length), an easting, a northing, an elevation, a zone number and a value. This information is stored in a "three-dimensional pilot points file" the formatting of which is identical to that of a normal pilot points file except for the requirement of an extra column of data which provides point *z* (i.e. elevation) coordinates. An example follows.

```
     1_1_a       683.013       7816.987    -5.0      1        3.435
   9_112_a     19179.963      11279.329    -5.0      1        7.876
   10_10_a      9903.684       5346.316    -5.0      1        7.323
   40_60_a     16007.016       5175.035    -5.0      1        9.543
   77_62_a     19705.221       -830.444    -5.0      1        1.345
  77_117_a     27369.545       3594.556    -5.0      1        2.675
     1_1_b       683.013       7816.987    -45.0     1        7.876
   9_112_b     19179.963      11279.329    -45.0     1        2.789
   10_10_b      9903.684       5346.316    -45.0     1        9.432
   40_60_b     16007.016       5175.035    -45.0     1       10.456
   77_62_b     19705.221       -830.444    -45.0     1        5.632
  77_117_b     27369.545       3594.556    -45.0     2        2.345
     1_1_c       683.013       7816.987    -220.0    2        1.342
   10_10_c      9903.684       5346.316    -220.0    2        9.542
  77_117_c     27369.545       3594.556    -220.0    2        4.321
```

Part of a three-dimensional pilot points file.

Columns in a three-dimensional pilot points file are as follows.

- The first column contains pilot points identifiers; these must be 10 characters or less in length.
- The second, third and fourth columns contain pilot point eastings, northings and elevations respectively.
- The fifth column contains the zone number to which each pilot point pertains.
- The last column contains the parameter value associated with each pilot point.

Three-Dimensional Geostatistical Structures

Like PPK2FAC, PPK2FAC3D prompts for the name of a structure file. It reads specifications for geostatistical structures from this file. Each geostatistical structure is comprised of one or more variograms. If three-dimensional kriging is to be carried out, variograms cited by a referenced geostatistical structure must be three-dimensional variograms. Note that a geostatistical structure can cite more than one variogram.

Three-dimensional variograms are illustrated in the figure below, which shows part of a structure file.

```
STRUCTURE struct1
  NUGGET 0.0
  TRANSFORM log
  MEAN 1
  NUMVARIOGRAM  1
#  VARIOGRAM var2c 0.6
  VARIOGRAM var2 0.3
END STRUCTURE

VARIOGRAM var2
  VARTYPE 2
  ANG1 0
  ANG2 0
  ANG3 0.0
  A_HMAX 50000
  A_HMIN 50000
  A_VERT 5
END VARIOGRAM

VARIOGRAM var2c
  VARTYPE 2
  ANG1 60
  ANG2 30
  ANG3 0.0
  A_HMAX 10000
  A_HMIN 10000
  A_VERT 30
END VARIOGRAM
```

Part of a structure file featuring three-dimensional variograms.

A three-dimensional variogram requires specification of seven variables. These are the VARTYPE, A, ANG1, ANG2, ANG3, A_HMAX, A_HMIN and A_VERT variables.

As for the two dimensional variogram, VARTYPE specifies the variogram type. This can be spherical, exponential, Gaussian or power, these being indicated by VARTYPE values of 1, 2, 3 and 4 respectively.

As for the two-dimensional variogram, A is the value of *a* in the variogram equation (this being the power in the case of the power variogram). See equations (5.1) to (5.4) of Part A of this manual; *a* is related to the range of the variogram. However for a three-dimensional variogram, three A values must be supplied, these being denoted as A_HMAX, A_HMIN and A_VERT. These are the values of A in the directions of maximum horizontal range, minimum horizontal range and vertically respectively.

The angles (in degrees) ANG1, ANG2 and ANG3 define geometric anisotropy of the three-dimensional variogram. Their role are as follows.

- ANG1 defines the angle between north and the direction of maximum horizontal anisotropy in degrees clockwise;

- ANG2 defines the plunge of the direction of maximum anisotropy, that is the angle (positive downwards) between horizontal (in the direction defined by ANG1) and the direction of actual maximum anisotropy;

- To quote Deutsch and Journel (1998) - "The third rotation angle ANG3 leaves the principal direction, defined by ANG1 and ANG2, unchanged. The two directions orthogonal to that principal direction are rotated clockwise relative to the principal direction when looking toward the origin." In the vast majority of cases ANG3 should be set to zero.

Any variogram appearing in a structure file must be either a two-dimensional or three-dimensional type. That is, it must provide values for ANG[1-3], A_HMAX, A_HMIN and A_VERT but not A, ANIS and BEARING; or vice versa. However any geostatistical structure used for three-dimensional kriging must cite only three-dimensional variograms. If these conditions are violated, PPK2FAC3D will cease execution with an appropriate error message.

A Three-Dimensional Grid Specification File

Optionally, PPK2FAC3D obtains MODFLOW surface and layer elevations from a three-dimensional grid specification file. This file type constitutes an expansion of the usual two-dimensional grid specification file; hence the latter is a subset of the former. The name of one or the other of these file types must be supplied to PPK2FAC3D when it asks for the name of a grid specification file. It will soon establish which type of file it is reading; if it finds itself reading a two-dimensional grid specification file it will obtain grid layering information from elsewhere (see below).

The figure below presents the format of a three-dimensional grid specification file.

```
nrow ncol nlay
e0 n0 rotation
(delr(icol), icol=1,ncol)
(delc(irow),irow=1,nrow)
spectype
if spectype equals 0
surface_elev
(thick(ilay),ilay=1,nlay)
or if spectype equals 1
array_filename_for_top_elevation
array_filename_for_layer_1_bottom_elevation
array_filename_for_layer_2_bottom_elevation
.
array_filename_for_layer_nlay_bottom_elevation
```

Specifications of a three-dimensional grid specification file.

SPECTYPE is an integer. If it is supplied as zero, then the following line must contain one real number, this being the (assumed uniform) elevation of the top of the model. Following that, starting on the next line and wrapping if necessary, must be an array of NLAY entries containing the (assumed uniform) thickness of each model layer.

If SPECTYPE is supplied as 1, then each of the following NLAY+1 lines of the three-dimensional grid specification file must contain the name of a file that holds a two-dimensional MODFLOW-compatible array. The first of these files must contain the cell-by-cell elevation of the surface of the model; files named on subsequent lines must contain cell-by-cell elevations of the bottoms of respective model layers, starting at 1 and finishing at NLAY. In accordance with the usual protocol of the Groundwater Data Utilities suite, these arrays must be preceded in their files by a number-of-columns, number-of-rows header if COLROW is set to "yes" in file *settings.fig* which must reside in the directory from which PPK2FAC3D is run.

Running PPK2FAC3D

As is standard for all other members of the Groundwater Data Utility suite, the user can backtrack to previous screen prompts at any stage of PPK2FAC3D execution by responding to the current prompt with "e" or "E" followed by <Enter>.

Upon commencement of execution PPK2FAC3D looks for a settings file named *settings.fig* in the current working directory. If it does not find one, it assumes a COLROW setting of "no". (The DATE setting is not required.)

PPK2FAC3D next prompts for the name of a grid specification file:

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

If the grid specification file is the two-dimensional type (see above), PPK2FAC3D then prompts:

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

It then asks:

```
Enter filename base of layer bottom elevation array files:
```

Suppose that the user responds to this prompt with the string "bottom". Then PPK2FAC3D will attempt to read two-dimensional arrays of model layer bottom elevations from files *bottom0.ref*, *bottom1.ref....bottomN.ref* where *N* is the number of layers in the model. Note the following.

- If COLROW is specified as "yes" in *settings.fig* then a number-of-columns, number-of-rows header is expected in each of these array files.

- The first array, namely that with an index of "0", must provide elevations of the top of layer 1.

PPK2FAC3D reads the above arrays so that it can calculate the elevation of the centre of each grid cell. Note that layer bottom elevation arrays can be easily extracted from a MODFLOW "discretization file" using the MOD2ARRAY utility. Note also that PPK2FAC3D will not ask for the names of these files if it can obtain layer elevation information from a three-dimensional grid file and files cited therein.

Next PPK2FAC3D prompts for the name of a three-dimensional pilot points file. Specifications for this type of file have been provided above.

PPK2FAC3D's next task is to ascertain the distribution of zones within the model domain. This is provided through a series of integer arrays, one for each model layer. However the user has the alternative option of assigning a single number to the entire model grid. Prompts are as follows:

```
Enter filename base of layer zonal integer array files.
Press <Enter> if entire grid belongs to single zone:
Enter zone number for entire grid:
```

If the user responds to the second of the above prompts simply by pressing the <Enter> key, then every cell in the entire model domain is assigned the same zone number, that number being provided in response to the third of the above prompts. Alternatively, suppose that a filename base of "intzone" is supplied in response to the second of the above prompts. Then PPK2FAC3D will attempt to read *N* two-dimensional integer arrays from *N* files named *intzone1.inf*, *intzone2.inf....intzoneN.inf*. If COLROW is supplied as "yes" in the settings file *settings.fig*, then each of these files must possess a number-of-columns, number-of-rows header.

Next PPK2FAC3D prompts for the name of a structure file. The prompt is:

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

For each zone within the model domain (these being identified through their citation in previously-supplied layer-specific zonal integer arrays, or as a single zone number characterizing the entire model domain), PPK2FAC3D asks:

```
Enter structure name for zone with integer value of 1:
Perform simple or ordinary kriging [s/o]:
Enter search radius in maximum horizontal elongation dirn:
Enter search radius in minimum horizontal elongation dirn:
Enter search radius in vertical dirn:
Enter minimum number of points to use for interpolation:
```

```
Enter maximum number of pilot points to use for interpolation:
```

Finally it asks for the name of the file in which it must store the kriging factors which it calculates:

```
Enter name for interpolation factor file:
Is this a formatted or unformatted file? [f/u]:
```

It then computes kriging factors, writes the kriging factor file, and ceases execution.

Uses of PPK2FAC3D

PPK2FAC3D is used in conjunction with the FAC2REAL3D utility. The latter reads kriging factors computed by PPK2FAC3D, as well as a three-dimensional pilot points file. It then undertakes three-dimensional interpolation to the centres of cells of a three-dimensional MODFLOW grid.

See Also

PPK2FAC, PPK2FAC2, PPK2FAC3 and FAC2REAL3D.

Reference

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

PPK2FACR

Function of PPK2FACR

The role of PPK2FACR is very similar to that of PPK2FAC, PPK2FAC1 and PPK2FACF in that it computes kriging factors. However, in contrast to the above programs, PPK2FACR was build for use with the RSM (Regional Simulation Model) developed by the South Florida Water Management District. Like other members of the PPK2FAC family, it also generates regularisation data for optional later inclusion in a PEST input dataset. However in doing this it adopts the same protocols as used by PPK2FAC1 rather than PPK2FAC. Hence PPKREG1 (rather than PPKREG) must be employed to add this information to a PEST control file.

Using PPK2FACR

Use of PPK2FACR is very similar to that of other members of its suite; hence only differences between it and those other members are discussed herein.

PPK2FACR commences execution with the prompt:-

```
Enter name of GMS two-dimensional mesh file:
```

This is the type of mesh file employed by the RMS model (as well as the GMS graphical user interface). An example of a small such file is provided below.

```
MESH2D
E3T     1    1     6     2     1
E3T     2    2     7     3     1
E3T     3    3     8     4     1
E3T     4    5    10     6     1
E3T     5    6    11     7     1
E3T     6    7    12     8     1
E3T     7    9    14    10     1
E3T     8   10    15    11     1
E3T     9   11    16    12     1
E3T    10    1     5     6     1
E3T    11    2     6     7     1
E3T    12    3     7     8     1
E3T    13    5     9    10     1
E3T    14    6    10    11     1
E3T    15    7    11    12     1
E3T    16    9    13    14     1
E3T    17   10    14    15     1
E3T    18   11    15    16     1
ND  1       0.000      15000.000 0.
ND  2    5000.000      15000.000 0.
ND  3   10000.000      15000.000 0.
ND  4   15000.000      15000.000 0.
ND  5       0.000      10000.000 0.
ND  6    5000.000      10000.000 0.
ND  7   10000.000      10000.000 0.
ND  8   15000.000      10000.000 0.
ND  9       0.000       5000.000 0.
ND 10    5000.000       5000.000 0.
ND 11   10000.000       5000.000 0.
ND 12   15000.000       5000.000 0.
ND 13       0.000          0.000 0.
ND 14    5000.000          0.000 0.
ND 15   10000.000          0.000 0.
ND 16   15000.000          0.000 0.
```

A two-dimensional mesh file of the type employed by the RSM model.

Next PPK2FACR prompts for the name of a pilot points file:-

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

and then, as for other members of this family, it asks:-

```
Enter minimum allowable points separation:
```

If any two pilot points are closer together than the separation supplied in response to this prompt, PPK2FACR will cease execution with an appropriate error message. Note that "zero" is a suitable response to this prompt.

PPK2FACR next asks:-

```
Read zonal index file? (y/n):
```

and, if the answer is "y":-

```
Enter name of zonal index file:
```

The latter file must employ RSM "index file" format, which supplies a value to every element in the mesh. The assignment of values to elements in this fashion constitutes

a means of mesh zonation. As is discussed below, different pilot points can be assigned to different zones; interpolation from pilot points to mesh elements within each zone is then undertaken separately so that inter-zonal property discontinuities can be introduced. The figure below shows an example of a zonal index file. Data must be comprised of integers, one for each element of the mesh. These must be arranged in order of increasing element index number.

```
DATASET
OBJTYPE "network"
BEGSCL
ND 18
NAME "segment index"
TS 0 0.0
1
1
1
2
1
1
2
2
1
2
1
1
2
2
1
2
2
2
```

A zonal index file.

Alternatively, if the user requests that a zonal index file not be read, PPK2FACR generates kriging factors for the entire mesh from all points nominated in the pilot points file.

Next PPK2FACR prompts for the name of a structure file.

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

This file contains definitions of one or more geostatistical structures on which computation of kriging factors will be based. See documentation of PPK2FAC for further details.

Next, for each zone defined in the zonal index file, PPKFACR issues the following series of prompts:-

```
For zone characterised by integer value of 1:-
  Enter structure name (blank if no interpolation for this zone):
  Perform simple or ordinary kriging [s/o]:
  Enter search radius:
  Enter minimum number of pilot points to use for interpolation:
  Enter maximum number of pilot points to use for interpolation:
```

See documentation of PPKFAC for more details.

Finally PPK2FACR prompts for the name of the kriging factors file which it must write and for its formatted/unformatted (i.e. binary) status:-

```
Enter name for interpolation factor file:
Is this a formatted or unformatted file? [f/u]:
```

and finally for the name of a file in which it should store regularisation data:-

```
Enter name for regularisation information file:
```

It then computes and stores kriging factors, and writes information to the regularisation information file. Kriging factors are subsequently used by the FAC2RSM utility in carrying out the actual spatial interpolation from pilot points to mesh elements. The regularisation information file is available for the use of the PPKREG1 utility in adding regularisation prior information equations to a PEST control file.

In computing interpolation factors PPK2FACR assumes that interpolation takes place from pilot points to the centroid of each mesh element. PPK2FACR computes the locations of these centroids itself.

Uses of PPK2FACR

PPK2FACR is a vital component of pilot points parameterisation of an RSM model. If calibration is undertaken using PEST, PEST assigns values to these pilot points. Interpolation from them to mesh elements is undertaken by FAC2RSM on the basis of kriging factors calculated by PPK2FACR. Calculation of kriging factors can be far more time-consuming than carrying out the actual interpolation. By computing these ahead of the interpolation process, this step can be omitted from the model that is run many times by PEST during the calibration process, thereby resulting in a lowering of the time required for completion of the PEST run.

See Also

See also FAC2RSM, RSM2SRF and RDAT2TAB.

PPK2FAC_FEFL

Function of PPK2FAC_FEFL

PPK2FAC_FEFL performs a similar role to that of programs PPK2FAC, PPK2FACF, PPK2FACR, etc in that it generates kriging factors through which spatial interpolation is undertaken from pilot points to model elements. PPK2FAC_FEFL performs this role for the FEFLOW finite element model. As well as generating kriging factors, it also generates information that can be employed for subsequent introduction of regularisation to a PEST control file in which pilot-point-based parameters are featured. Kriging factors computed by PPK2FAC_FEFL are used by PPK2FAC_FEFL's sister program FAC2FEFL which carries out the actual spatial interpolation from pilot points to the FEFLOW mesh. FAC2FEFL often comprises part of a "model", encapsulated in a batch or script file, which is run many times by PEST in the course of optimising pilot point (and other) parameter values.

Using PPK2FAC_FEFL

Prompts and User Responses

Use of PPK2FAC_FEFL is not significantly different from that of other members of the PPK2FAC family. Hence only a brief description of its operations (concentrating on differences between this program and its kindred programs) is presented herein. See other sections of this manual (particularly documentation of PPK2FAC) for further details.

Like all other programs of the Groundwater Data Utility suite, PPK2FAC_FEFL allows a user to backtrack from a current prompt to a previous prompt to rectify any mistakes that he/she may have made in responding to that prompt. This is achieved through responding to the current prompt with "e" (for "Escape") followed by <Enter>.

PPK2FAC_FEFL screen output, and user responses to PPK2FAC_FEFL prompts, for a typical PPK2FAC_FEFL run are shown below. (User responses are depicted in bold italicised type.)

```
Program PPK2FAC_FEFL calculates point-to-element factors by which kriging is
   undertaken from a set of pilot points to a FEFLOW mesh.

 Enter name of FEFLOW FEM file for current project: project.fem

 Enter name of FEFLOW element property file: em.dat
  - data for 92829 elements read from element property file em.dat

 Enter name of pilot points file: pp.dat
  - data for 27 pilot points read from pilot points file pp.dat

 Enter minimum allowable points separation: 0

 Enter name of structure file: struct.txt
```

```
The following zones have been detected in the element property file:-

   For zone characterised by integer value of 1:-
   Enter structure name (blank if no interpolation for this zone): <Enter>

   For zone characterised by integer value of 2:-
   Enter structure name (blank if no interpolation for this zone): struct1
   Perform simple or ordinary kriging [s/o]: o
   Enter search radius: 1e20
   Enter minimum number of pilot points to use for interpolation: 3
   Enter maximum number of pilot points to use for interpolation: 8

   For zone characterised by integer value of 3:-
   Enter structure name (blank if no interpolation for this zone): <Enter>

Enter name for interpolation factor file: factors.dat
Is this a formatted or unformatted file? [f/u]: f

Enter name for regularisation information file: reg.dat

Carrying out interpolation for zone 2....

  Number of pilot points for this zone     =    27
  Mean data value for these pilot points   =   14.000
  Data standard deviation for these points =   7.7889
  Working....
  No. of elements for which factors were calculated       =  30943
  No. of elements beyond search radius of any pilot point  =      0

 - kriging factors written to file factors.dat
 - regularisation information written to file reg.dat
```

PPK2FAC_FEFL commences operations by opening the FEFLOW "*fem*" input file for the current model; it reads from this file the number of elements in the model domain.

PPK2FAC_FEFL next reads an "element property file". This is a file that must be exported from FEFLOW by the user. See the next subsection for details. The first few lines of a typical element property file are shown in the figure below.

ELEM	LAYER	X	Y	Z	F
1	1	73541.8444896667	8460384.0101699997	20.57783333	1.00
2	1	133405.5944896667	8450245.7393366657	105.09966667	1.00
3	1	138278.2507396667	8369087.4893366667	82.29783333	1.00
4	1	117906.1101146667	8363388.8018366667	111.38166667	1.00
5	1	133744.0736563334	8394474.3643366657	96.85883333	1.00
6	1	144707.7559480000	8382273.5310033327	66.54083333	1.00
7	1	141873.8705313334	8372043.3435033327	80.98366667	1.00
8	1	111523.2351146667	8375981.8018366667	204.01000000	1.00
9	1	103691.3965730000	8385102.5726699997	131.89800000	1.00
10	1	207472.6361563333	8303504.1351699997	171.03616667	1.00
11	1	109153.3601146667	8380700.6976699997	232.00733333	1.00
12	1	84339.4903230000	8461949.8018366657	44.89433333	1.00
13	1	110151.6674063334	8377364.5518366667	206.97566667	1.00
14	1	106335.4851146667	8382501.0935033327	183.28333333	1.00
15	1	101455.3184480000	8468199.2393366657	61.68633333	2.00
16	1	101939.1049063334	8396911.2601699997	131.61033333	2.00
17	1	107046.1205313334	8380231.2810033327	194.08133333	2.00
18	1	212998.8340730000	8319898.6976699997	150.70183333	2.00
19	1	92884.0007396667	8458993.3435033336	39.23600000	2.00
20	1	105327.7194896667	8428316.5726699997	78.42116667	2.00
21	1	242948.7299063333	8391365.7185033336	187.82216667	2.00
28	1	114709.1153230000	8418550.1976699997	60.00183333	2.00

Part of an element property file.

The first line of an element property file contains the column headers depicted in the above figure. Then follows data arranged in columns. Within each row of this file the first column contains the number of a node. These are arranged in increasing sequential order; all nodes within the mesh must be represented (see below). The easting, northing and elevation of each node follow. Then follows the material property value associated with that node. PPF2FAC_FEFL actually interprets entries in this column as zone numbers, in spite of the fact that these are actually real numbers (and are recorded by FEFLOW as material property values). Before exporting this file, it is the user's responsibility to supply values for the pertinent material property to all elements of the FEFLOW model, with these values being actually surrogate zone values. Normally only one, or a small number of zones, need be represented in a model. PPK2FAC_FEFL obtains a zone number from a property value through approximating that property value by its nearest integer.

After this file is exported, there may not be any need to alter pertinent material property values within the FEFLOW interface to something other than integers which represent zone numbers. As will be explained below (and as is also explained in documentation to program FAC2FEFL) if, for a particular material property, all elements within a model are to be assigned values on the basis of interpolation from pilot points, replacement of zone values by pertinent property values will take place at the level of the FEFLOW "*fem*" input file.

PPK2FAC_FEFL obtains the following information from the element property file:-

1. the easting and northing of each finite element mesh centroid of the current model (it is to these coordinates that spatial interpolation takes place from pilot points);

2. the zone to which each element belongs.

As is discussed in documentation of the PPK2FAC utility, different pilot points, and different geostatistical structures on the basis of which kriging factors are calculated, can be assigned to different zones. Alternatively, a zone may have no pilot points and no geostatistical structure assigned to it; hence interpolation will not take place from pilot points to elements within that zone. In the latter case, material property values will not be altered from those already assigned to elements within the zone when FAC2FEFL alters a FEFLOW "*fem*" file prior to a FEFLOW model run.

Other information sought from the user by PPK2FAC_FEFL is similar to that sought by other PPK2FAC programs. The reader is referred to documentation of those programs for details.

Exporting an Element Property File

An element property file is generated by exporting flow material properties from FEFLOW in "*dat*" file format. This produces a tab-delimited ASCII file with the following fields (see the above example):-

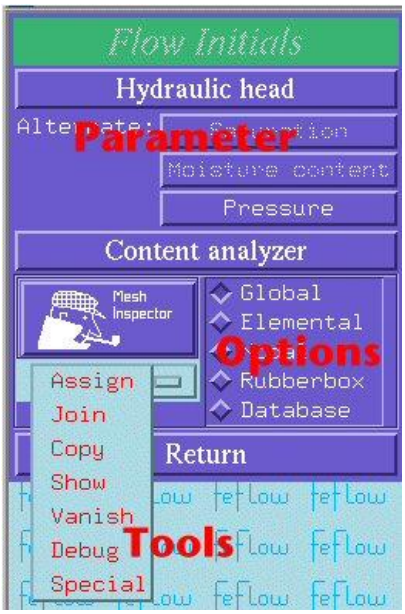
- element number,
- layer number,
- X coordinate,
- Y coordinate,
- Z coordinate, and
- F value (i.e. property value, for example K_{xx}).

To export this file, the following steps are required.

1. Open the FEM file pertaining to the current model using the FEFLOW graphical user interface.

2. Select "Edit", then "Edit problem attributes", then "Flow data", then "Flow materials" from respective FEFLOW menus.

3. Select "Special" from the "Tools" list (see below).



FEFLOW menu system for exporting of a element data file.

4. Select the property to be exported (e.g. "Conductivity [Kxx]"). The FEFLOW "Data View, Operation and Export" dialog box then opens.

5. From the "Save as plot file…" area of this box, change "Save nodal materials as points" to "Save centre materials as points".

6. Select "ASCII Databases (XYZF) (*.dat)" as the file type option, and provide a name for the file.

7. Click on "yes" when asked by FEFLOW whether to "Export quantities for all slices".

Non-Interpolation

There are two reasons why interpolation factors will not be assigned to a particular element of the FEFLOW mesh. In particular,

- if an element is assigned to a zone to which no pilot points are assigned, or

- if an element's location is such that it is separated by more than one search radius from all pilot points assigned to its zone,

then no interpolation from pilot points can take place to that element.

In contrast to other interpolation programs provided with the Groundwater Data Utility suite, when FAC2FEFL undertakes spatial interpolation to the FEFLOW mesh on the basis of kriging factors calculated by PPK2FAC_FEFL, it does not ask the user what value should be assigned to elements for which no interpolation factors have been calculated. Instead it simply retains property values that are already assigned by FEFLOW to those elements when it writes a new FEFLOW "*fem*" input file through

modification of an existing one. Thus elements (including entire layers) for which no pilot point parameter definition is desired can all be assigned to a single zone to which neither pilot points nor a geostatistical structure is assigned.

The user should be very careful, however, that lack of kriging factor computation for certain elements is not an unwanted outcome of providing PPK2FAC_FEFL with an interpolation search radius which is too small. In reporting the results of the kriging factor computation process for each zone, PPK2FAC_FEFL writes to the screen the number of elements within each such zone for which kriging factors could not be computed for this reason. If this number is not zero, it is flagged with a "WARNING" string. Under most circumstances, on receiving such a warning, the user should re-run PPK2FAC_FEFL, asking for a larger search radius. (It is normally good practice to provide an effectively infinite search radius of 1.0E20, and limit the number of pilot points involved in spatial interpolation to any element through an appropriate response to the "enter maximum number of pilot points to use for interpolation" prompt.

Model Layering

As for other programs of the PPK2FAC family, kriging factors implement only horizontal interpolation. Nevertheless, the element property file should cite all elements within a model mesh, irrespective of the layer to which each element is assigned. In many cases it will be desirable that all model layers comprising a particular hydrostratigraphic unit be assigned the same hydraulic property value. In this case all elements within any vertical column of that hydrostratigraphic unit should be assigned to the same zone; because elements with the same easting and northing will automatically be assigned the same kriging factors from the same set of pilot points, this will guarantee that the interpolated material property is the same for all such cells.

In other modelling circumstances it may be desired that different properties be assigned to cells in different model layers. This is easily achieved by assigning elements in different layers to different zones. Pilot points must then be assigned separately to each such zone within the pilot point file whose name is supplied to PPK2FAC_FEFL. There is no reason why pilot points assigned to different layers cannot be placed at the same horizontal locations. (This can be achieved through appropriate cutting and pasting in the pilot points file.) However pilot point names must be different for the different layers (a discriminatory pilot point name prefix will easily achieve this).

Regularisation Data

As for other members of the PPK2FAC family, PPK2FAC_FEFL writes a regularisation information file. This can be used by the PPKREG1 utility for adding regularisation prior information to a PEST control file. (Alternatively, the PEST ADDREG1 utility can be used for the same purpose.)

Uses of PPK2FAC_FEFL

PPK2FAC_FEFL is used, in conjunction with FAC2FEFL, for implementation of pilot-point-based parameterisation of a FEFLOW model. Though the manner in which kriging factors are calculated is the same for PPK2FAC_FEFL as it is for other members of the PPK2FAC family, there is an important difference where multi-layered models are employed, in that all pilot points employed for spatial interpolation to all model layers must be cited in the same pilot points file. Where there is no vertical variation of hydraulic properties between a number of juxtaposed model layers, elements within all pertinent layers should be assigned to the same zone as that to which pilot points belonging to the hydrostratigraphic unit encompassing these layers are assigned. However where different layers, or different groups of layers, must be assigned different values through interpolation from different sets of pilot points, the layers pertaining to different hydrostratigraphic units must be assigned to different zones; all pilot points for all zones must be then be represented in the single pilot points file read by PPK2FAC_FEFL (and subsequently by FAC2FEFL).

The DAR2SMP utility provides a means through which FEFLOW outputs can be rapidly processed for inclusion in a PEST-based calibration process.

See Also

See also PPK2FAC, PPK2FAC1, PPK2FACF, PPK2FACR, FAC2FEFL, PPKREG1 and DAR2SMP.

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. 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

See Also

See also 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.

Using 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 (i.e. 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.

PPSAMP

Function of PPSAMP

PPSAMP assists in calibration-constrained Monte Carlo analysis, and is designed to be used in conjunction with the PEST utility PNULPAR. It obtains random values at pilot point locations through sampling stochastic fields (generated, for example, using the FIELDGEN utility documented elsewhere in this manual); sampling can be either direct or of the least squares type. It then generates a series of "difference fields", each of these being comprised of differences, computed at every active cell in the grid, between an original stochastic field and a field obtained through interpolation between sampled pilot point values (using the same kriging-based interpolation algorithm as that employed by PPK2FAC and FAC2REAL). If this difference field (which normally represents variation of hydraulic properties on a small spatial scale) lies mostly within the calibration null space, then it can be added to a pilot-point-based random field computed by PNULPAR to supply the hydraulic property detail that is lost through the use of pilot points. The summed field can then be "warped back into calibration" by PEST so that model outputs replicate historical system behaviour.

Using PPSAMP

Background

Use of PPSAMP is predicated on the assumption that a calibrated model exists, and that at least some of its parameters are based on pilot points. It is assumed that FAC2REAL is employed for spatial interpolation of values assigned to pilot points to the model grid, and thus that a PPK2FAC-produced "factor file" is present.

As is discussed in papers such Moore and Doherty (2005) and Moore and Doherty (2006), some form of regularisation is required to achieve a calibrated model. This results in parameter fields which are smoothed or simplified versions of hydraulic property reality. Predictions (especially predictions that are sensitive to the level of system detail that cannot be represented in a calibrated model), are thus likely to be in error. Quantification of this potential error should be an important part of modelling practice if modelling is to be undertaken with integrity.

One method through which predictive uncertainty (which, in the current context will be used interchangeably with "possible predictive error") can be quantified is through Monte Carlo analysis. Using this technique many different stochastic fields are generated, each of which is a realisation of the hydraulic property detail that may exist at a study site, but which cannot be represented uniquely in a calibrated model because this detail is simply beyond the reach of the calibration process. A particular prediction can then be made with all such fields; the variability of that prediction between hydraulic property realisations can then be used to assess model predictive uncertainty.

Where hydraulic properties employed by a model are not constrained by the necessity for model outputs to match field measurements under calibration conditions, stochastic analysis is easily undertaken. In this case it is comprised simply of running the model many times using different hydraulic property fields, and collecting outputs of interest generated during each model run for stochastic analysis. However where each such field must be such that the model remains in a calibrated state, the generation of hydraulic property fields becomes a far more difficult process. PPSAMP is designed to assist in this process.

Calibration-constrained Monte Carlo analysis can be undertaken with the help of the PEST PNULPAR utility. Using this utility, the calibrated parameter set is subtracted from different sets of random parameters (generated, for exampling using the PEST RANDPAR utility). These differences are then projected onto the calibration null space; the projected differences are then added to the calibrated parameter set to obtain different sets of random parameters, all of which would calibrate (or nearly calibrate) the model if it were linear. If the model is nonlinear (as most are), "warping" of this new parameter set using one PEST iteration based on pre-calculated sensitivities (updated with a Broyden Jacobian upgrade) can often bring the model back into a calibrated state within a few model runs. Thus a suite of stochastic parameter fields, all of which calibrate the model, can be rapidly generated. These can then be used for Monte Carlo analysis.

This same procedure can be employed to generate calibration-constrained random parameter fields where parameters pertain to pilot points, such as are often employed for parameterization of a groundwater model. However as model cell property values between pilot points are normally computed from pilot point values using a smooth interpolator (such as kriging), such fields lack the fine detail that is normally present in stochastic fields, such as those generated using the FIELDGEN utility. This "fine detail" which "fits between pilot points" must be restored to these fields prior to undertaking probabilistic analysis of model outputs, for to the extent that any such output is dependent on fine spatial detail, variability of that prediction will be underestimated through its omission.

What PPSAMP Does

PPSAMP has two primary functions.

The first function of PPSAMP is to obtain random values of pilot point parameters. It does this by sampling real arrays of stochastic hydraulic properties. These arrays can be generated using, for example, the FIELDGEN utility which assumes a multi-Gaussian hydraulic property distribution; or they may be generated with user-supplied software whose basis for achieving hydraulic property realisations may be purely stochastic, or a combination of stochastic and physical/chemical process simulation (for example a sedimentation model).

PPSAMP stochastic field sampling can be undertaken through either of two methods. If the "direct method" is employed, the value assigned to a pilot point is simply the real array value at the location of that point. If the pilot point does not lie exactly at a cell centre, then bilinear interpolation is undertaken from the real array to the location

of the point. Alternatively, sampling can be of the least squares type. In this case pilot point values are computed in such a manner that differences between field values interpolated between pilot points (using kriging) and stochastic field values pertaining to the original real array are minimized over the model grid in the least squares sense. In this minimisation process, differences are computed at every grid cell and weighted equally. Limited testing to date (see Moore, 2006) suggests that the latter method results in a "difference field" that is more likely to lie within the calibration null space. Thus addition of this difference field to a pilot point field is less likely to uncalibrate an already-calibrated model.

PPSAMP's second task is to build this difference field, this being assigned to the elements of a "difference real array". In this real array the above-mentioned difference between the pilot-point-interpolated field and the stochastic field from which pilot point values are sampled is recorded on a cell-by-cell basis.

Calibration Constrained Stochastic Analysis

The use of null space projection as a device for generating parameter sets which respect real world property variability, and which also satisfy calibration constraints, is described in the documentation of the PEST PNULPAR utility. PPSAMP extends the use of this methodology to Monte Carlo analysis based on property fields which allow cell-by-cell variability, such as those produced by the FIELDGEN utility, or any other utility that generates physically plausible fields and stores them in MODFLOW real-array format.

The first step in implementing this process is achieved through providing random values to pilot-point-based parameters by sampling these values from stochastic property arrays. Then, for each sampled stochastic array, a "hydraulic property difference array" is generated by subtracting from the original hydraulic property array another array produced through interpolation of pilot-point-sampled values to model cells represented in the array. Supposedly the interpolation mechanism through which model cell values are informed by pilot point values is the same as that used during the calibration process through which PEST-estimated pilot point values are interpolated to the cells of the model grid. Because this difference field will, in general, represent "fine system detail" (the greater the pilot point density, the more likely this is to be the case), its presence or absence should not have a large effect on the calibrated status of the model; that is to say, its projection onto the calibration solution space should not be large. (If it is indeed large, this is a sign that more pilot points should be employed in the calibration process.)

Like the PEST RANDPAR utility, PPSAMP produces a set of parameter value files. However where PPSAMP is employed, at least some of the parameters contained within these files pertain to pilot point parameters. These parameter value files can then be employed in conjunction with the PNULPAR utility to generate another set of parameter value files which "almost calibrate" the model; hopefully, only one PEST iteration on the basis of pre-calculated sensitivities will return the model to a calibrated state if any of these parameter sets are used by it. However, as stated above, hydraulic property real arrays computed from these parameter sets by spatial

interpolation between pertinent pilot points will lack the cell-by-cell detail of stochastic property fields. This fine detail must thus be added to them. This is achieved by adding the "hydraulic property difference array" computed by PPSAMP to the real array computed through interpolation from the PNULPAR-computed pilot point parameters (before re-calibration is undertaken). Recall that the hydraulic property difference array was computed by PPSAMP on the basis of the same stochastic field from which pilot point values were sampled before null space processing of the latter by PNULPAR; it is important that this pairing of stochastic fields, pilot point parameters and difference fields be respected when handling multiple parameter sets, as is normally done in Monte Carlo analysis.

If a model calibrated by PEST employs pilot points for one or more property types in one or more model layers, then one or more commands of the following type will appear in the model batch or script file:-

```
fac2real < fac2real.in
```

(Note that a redirected keyboard input files employed with the FAC2REAL utility does not need to be named *fac2real.in*; it can be named whatever the user chooses.) Suppose that user responses contained in *fac2real.in* inform FAC2REAL that it must write its model-ready real array to a file named *array.ref*. Before undertaking calibration-constrained Monte Carlo analysis, this response should be altered so that the array is instead written to a different file named, for example, *dummy.ref*. Another command of the following type should then be added to the model batch or script file immediately after the above command:

```
twoarray < twoarray.in
```

If interpolation from pilot points to MODFLOW/MT3D arrays is based on a log variogram (as it should be for most hydraulic properties), then the responses contained in *twoarray.in* should instruct TWOARRAY to multiply *dummy.ref* by the pertinent PPSAMP-generated hydraulic property difference array (additions in the log domain become multiplication in the domain of natural numbers) to produce a file named *array.ref* which is then employed by the model. If interpolation is not based on a log variogram, then *array.ref* should be obtained by adding the difference array to *dummy.ref*. Thus the hydraulic property array received by the model has had its fine detail returned to it.

Running PPSAMP

Like all programs of the Groundwater Data Utilities, PPSAMP is supplied information through user responses to a series of prompts. If an incorrect response is supplied at any stage, the user can backtrack to the previous prompt through responding to the next prompt with "e" (for "escape") followed by the <Enter> key.

Like other programs of the Groundwater Data Utilities suite, PPSAMP will not run unless a file named *settings.fig* is present within the current working directory. As explained in Part A of this manual, this informs these programs of the date protocol which they must use, and whether or not a "number of columns, number of rows" header is expected on the first line of real and integer array files. The former is not

needed by PPSAMP, as it does not handle time-based data. However the latter is required.

PPSAMP commences execution with the prompt:-

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

in response to which the name of the appropriate file should be supplied. It then asks:-

```
Enter filename base of random field arrays:
```

Suppose that the response to this prompt is "*base*". Then PPSAMP will look for a series of files named *base1.ref, base2.ref, base3.ref* etc, each of which contains a formatted stochastic real array.

PPSAMP then prompts for the name of a pilot points file. This same pilot point file should have been employed by the model as a basis for parameterisation of one hydraulic property type for at least one layer. Hence a template file should be matched to this pilot point file. PPSAMP does not read this template file, for it reads the pilot points file itself. Furthermore, it reads only pilot point names and coordinates from this file.

PPSAMP's next prompt is:-

```
Enter parameter prefix:
```

As is the PPK2FAC and FAC2REAL convention, the names of pilot point parameters as estimated by PEST are obtained by prefixing pilot point names (which should be ten characters or less in length) with a suitable prefix. Armed with knowledge of this prefix, PPSAMP is able to link PEST parameters (as read later from parameter value files) to associated pilot points. It then prompts:-

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

This is the PPK2FAC-produced file through which hydraulic property values on a cell-by-cell basis as employed by MODFLOW/MT3D are computed from pilot point property values as estimated by PEST. It is assumed that this file is employed in conjunction with the above set of pilot points by the FAC2REAL utility (which undertakes this spatial interpolation from pilot points to model real arrays) when run as part of the model. (Note that use of the formatted option is recommended. Use of unformatted files, though marginally quicker to read, and requiring less storage space, brings with it certain obscure problems that may promulgate a chain of events culminating in a user sending me an email complaining of PPSAMP malperformance. The writer wishes to avoid such emails, as the user probably does as well.)

PPSAMP's next prompt is:-

```
Employ direct or least-squares sampling of stochastic fields?  [d/l]:
```

Direct sampling is quicker. However limited testing to date suggests that least-squares sampling may lead to a difference field that has a smaller projection onto the calibration solution space. However its computation is longer, and if there are many pilot points it is possible that this calculation may become unstable. Nevertheless, at

the time of writing, the latter option is suggested. Feedback on this would be welcome.

PPSAMP next asks:-

```
Write new parameter value files or modify existing ones?  [n/m]:
```

Two options are available here. The first (if the user responds to the above prompt with "n") is that a single parameter value file is supplied in response to the prompt:-

```
Enter name of an existing parameter value file:
```

PPSAMP identifies parameters within this file that correspond to pilot points in the nominated pilot points file, and then writes a series of parameter value files in which these parameters are provided with values sampled from stochastic real arrays. All other parameters remain unchanged. PPSAMP asks:

```
Enter filename base for new parameter value files:
```

If the response to this prompt is *base*, then PPSAMP will generate files named *base1.par, base2.par, base3.par* etc, where each such parameter value file is linked to a corresponding stochastic real array file by filename base numeric suffix.

Alternatively, if the user responds to the above prompt with "m", PPSAMP asks:-

```
Enter filename base of existing parameter value files:
```

Suppose that the user responds to the above prompt with the string "*base*". Then PPSAMP looks for existing parameter value files *base1.par, base2.par, base3.par* etc. There must be one such file corresponding to each stochastic real array file (linked by filename base numeric suffix). PPSAMP then places sampled values from each real array file into the corresponding parameter value file, leaving all parameters apart from the sampled pilot point parameters unchanged in those files. This allows PPSAMP to be used sequentially to sample (and calculate difference arrays for) different hydraulic property arrays employed by the same model. (Note that a different instance of TWOARRAY must be introduced to the model batch file for each such property array in the manner described above.)

Next PPSAMP asks:-

```
Enter filename base for random field difference arrays:
```

If the name *diff* is provided, then hydraulic property difference arrays will be written to files *diff1.ref, diff2.ref, diff3.ref* etc. Note that formatted array storage is assumed. Finally PPSAMP asks for a number to use in these arrays for cells to which no pilot point interpolation takes place (presumably inactive cells). The prompt is:-

```
Enter dummy value for inactive cells for these arrays:
```

in response to which an easily identified number such as 1.0E35 will be suitable on most occasions.

Having now acquired all of the information that it needs to run, PPSAMP undertakes its mission, recording its progress to the screen. If an error condition is encountered an appropriate message is written to the screen before PPSAMP ceases execution.

The following should be noted.

1. The use of pilot points does not preclude the use of zones. PPK2FAC allows pilot points to be employed in different user-defined zones, with no interpolation occurring across zone boundaries. If a zone is homogeneous, it need contain only a single pilot point.

2. Where FAC2REAL interpolation from pilot points to a model-compatible real array is based on more than one zone, then the transformation status of variograms governing the interpolation process in all zones must be the same. That is, all hydraulic property interpolation should be based on a log variogram, or on a variogram with which no log transformation is associated.

3. Where interpolation is log-based (as it normally is) the "hydraulic property difference array" in fact contains hydraulic property quotients. Use the LOGARRAY utility to convert these to differences (of log hydraulic properties) if desired.

4. As mentioned above, where more than one layer or property type is informed by pilot points, PPSAMP can be used sequentially to modify parameter value files and generate difference arrays pertaining to the different property types.

Uses of PPSAMP

An Example

To illustrate the use of PPSAMP in computing a suite of different stochastic fields which all calibrate a simple one layer model, a short description of its use is provided.

Suppose that you have set up a calibration problem in which the hydraulic conductivity variation within a single layer is estimated. Suppose further that the following applies.

1. Parameterisation of hydraulic conductivity takes place on the basis of pilot points.

2. PEST input files have been prepared in which Tikhonov regularisation of estimated hydraulic conductivity values is implemented. As described in the PEST manual, this requires that a target measurement objective function Φ^1_m be selected; PEST "aims for" this objective function as it adjusts parameter values, and will cease execution if the measurement objective function falls below it.

3. Calibration is implemented using the highly efficient SVD-assist scheme.

Suppose that the base parameter PEST control file is named *base.pst*, that NOPTMAX has been set to -1 or -2 in this file and that a Jacobian matrix (named *base.jco*) has then been computed. The next step is to generate a super parameter PEST control file using SVDAPREP. Suppose that the name given to this file is *super.pst*. Suppose also that the SVDAPREP default setting of 1 for the SVDA_SUPDERCALC variable has been accepted. Hence during the first iteration of

the SVD-assisted parameter estimation process, PEST computes super parameter derivatives from base parameter derivatives, this eliminating the need for model runs to be undertaken for this purpose.

Make sure that Broyden Jacobian updating of the Jacobian matrix is activated in *super.pst* by setting the JACUPDATE variable to 999. Then run PEST on the basis of *super.pst* to calibrate the model using the command:-

```
pest super
```

Now create a file named *base1.pst* in which initial parameter values are in fact optimised parameter values. Employ the PARREP utility as follows:-

```
parrep base.bpa base.pst base base1.pst
```

Now (on the assumption that NOPTMAX in *base1.pst* is still set to -1 or -2), run PEST to obtain a Jacobian file *base1.jco* computed on the basis of optimised parameters. The command is:-

```
pest base1
```

The next task is to generate a set of stochastic fields. Use FIELDGEN to do this, naming its output real array files *k_stoch1.ref*, *k_stoch2.ref*, *k_stoch3.ref* etc. Make sure that a log variogram is employed as the basis for random field generation, probably the same variogram that was previously employed for computation of kriging factors (by PPK2FAC) through which cell-based hydraulic conductivities are computed from pilot point conductivities.

Next PPSAMP should be employed for generation of a suite of parameter value files and corresponding hydraulic property difference array files. An example set of PPSAMP prompts and responses is shown below.

```
Enter name of grid specification file: rect.spc
 - grid specifications read from file rect.spc

Enter filename base of random field arrays: k_stoch

Enter name of pilot points file: hk.pts
 - data for 104 pilot points read from pilot points file hk.pts

Enter parameter prefix: k_

Enter name of interpolation factor file: factors.dat
Is this a formatted or unformatted file? [f/u]: f

Employ direct or least-squares sampling of stochastic fields?  [d/l]: l

Write new parameter value files or modify existing ones?  [n/m]: n
Enter name of an existing parameter value file: base1.par
Enter filename base for new parameter value files: k_stoch

Enter filename base for random field residual arrays: k_diff
Enter dummy value for inactive cells for these arrays: 1e35
```

PPDEF generates a series of parameter value files named *k_stoch1.par*, *k_stoch2.par*, *k_stoch3.par* etc containing (least squares) samples of the stochastic fields contained in files *k_stoch1.ref, k_stoch2.ref, k_stoch3.ref* etc. Hydraulic property difference arrays are named *k_diff1.ref*, *k_diff2.ref*, *k_diff3.ref*, etc.

Now that a series of parameter value files containing random values for pilot point parameters has been prepared, another set of parameter value files must be prepared based on null space projection of differences between these random parameter values and parameters that calibrate the model. For this the PEST utility PNULPAR must be employed. However before this can be done, another base PEST control file must be prepared in which PEST is instructed to run in estimation mode, and in which all regularisation prior information is removed. (This is a PNULPAR requirement; it eradicates confusion pertaining to what observations and prior information equations are employed for determination of the null space.) So *base1.pst* should be copied to *base2.pst* and the necessary editing of this file undertaken using a text editor. A corresponding JCO file can then be produced, without actually running PEST using the command:-

```
jco2jco base1 base2
```

Then run PNULPAR. An example set of prompts and responses is provided below.

```
Enter name of PEST control file: base2.pst
Does PEST control file contain calibrated parameter values? [y/n]: y

Enter number of dimensions of calibration solution space: 12
Would you like to store Q(1/2)X matrix in matrix file format? [y/n]: n

Enter filename base of existing parameter value files: k_stoch
Enter filename base for new parameter value files: k_stoch_new
```

We now have a suite of pilot point based parameters which, if our model was linear, would all calibrate the model. These are contained in the suite of parameter value files whose members are named *k_stoch_new1.par, k_stoch_new2.par, k_stoch_new3.par* etc. However our job is not yet complete. This is because:-

1. Our model is nonlinear. Therefore these new parameter sets will not necessarily calibrate the model as well as the set of pilot point parameters estimated through the calibration process contained in *base.bpa* (and *base1.par*).

2. Because our random parameters are pilot point parameters, the hydraulic property real arrays generated from these by spatial interpolation will be too smooth to be realistic realisations of a heterogeneous subsurface.

These problems can be rectified by undertaking the following steps.

1. Alter the model batch file *svdabatch.bat* so that the FAC2REAL command is replaced by commands to run FAC2REAL followed by TWOARRAY. Suppose that FAC2REAL is instructed to write a real array file named *hk.ref* which is then employed by the model. Alter the pertinent response in the redirected keyboard input file so that it writes a file named *dummy.ref* instead.

2. Prepare a TWOARRAY redirected keyboard input file which instructs TWOARRAY to multiply *dummy.ref* by an array named *diff.ref* to produce a file named *hk.ref*.

3. Add the command to delete *dummy.ref* to the top of the batch file to prevent repetitious reading of the same input file in the event of TWOARRAY execution failure.

4. Edit the super parameter control file *super.pst* as follows (see documentation of PNULPAR for more details):-

    a. In the "svd assist" section, alter "base.pst" to "base1.pst" and "base.jco" to "base1.jco". Thus PEST is asked to look for initial parameter values in *base1.pst* and to *base1.jco* for sensitivities on which to define super parameters and to compute derivatives of model outputs with respect to super parameters during its initial iteration.

    b. Set NOPTMAX to 1.

    c. Set RLAMDA1 to 1.0

    d. Set PHIRATSUF to 0.001

    e. Set RELPARMAX to 0.4

    f. Set JACUPDATE to 999.

    g. In the "regularisation" section set the target objective function PHIMLIM to about 70% of its previous value. Set PHIMACCEPT about 10% higher than this.

5. Now, for each stochastic field and corresponding parameter value file that was previously generated by PPSAMP (these being linked by the filename base suffix *n*), issue the following set of commands.

```
parrep k_stoch_newn.par base.pst base1.pst
copy k_diffn.ref diff.ref
pest super
```

6. After each PEST run (which should last for just one iteration, and which should require no computation of derivatives) copy *hk.ref* to a file of a suitable name, for example *hkn.ref*. The collection of these files will then constitute a set of stochastic fields which can be employed for exploring the variability of any model prediction.

Fixed Pilot Point Parameters

If you wish that the hydraulic properties at one or more locations within the model domain (for example the locations where pumping tests were carried out) are assigned the same values in all hydraulic property field realisations, this can be achieved through the following mechanism.

1. Place pilot points at the centres of cells whose values you wish to remain invariant from stochastic field to stochastic field.

2. Supply these conditioning values to FIELDGEN when generating stochastic fields.

3. Fix the parameters corresponding to these pilot points at their desired values during model calibration.

4. You can request either direct or least squares sampling of stochastic fields when employing PPSAMP. In the former case the value of pertinent pilot point parameters in PPSAMP-generated parameter value files will be unchanged from their fixed values (provided pilot points truly lie at the centres of model cells). In the latter case they will be altered; however when the difference array value is added to this altered value, hydraulic property values at pertinent cells of the final array should equal the desired cell values. These hydraulic property array cell values will be retained even after new parameter value files are generated by PNULPAR for, as described in the documentation to that program, where a parameter is fixed, the value in the parameter value file overrides that in the existing PEST control file. Hence alterations to its original value incurred through least squares sampling will still be exactly compensated for by pertinent cell values in each matching difference array.

Note that cell centre coordinates can be obtained using the GRID2PT utility.

See Also

See also PPK2FAC, FAC2REAL, FIELDGEN, TWOARRAY.

References

Moore, C., (2006), The use of regularized inversion in groundwater model calibration and prediction uncertainty analysis. *PhD Thesis*, University of Queensland, Australia.

Moore, C. and Doherty, J., 2005. The role of the calibration process in reducing model predictive error. *Water Resources Research*. Vol 41, No 5. W05050.

Moore, C., Doherty, J., 2006. The cost of uniqueness in groundwater model calibration. *Advances in Water Resources*. Volume 29, Issue 4, April, pages 605 – 623.

Acknowledgement

The writing of this utility was supported by a contract with Boise State University under USEPA Grant X-96004601-0. I wish to express my gratitude for this support.

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.

Using 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. 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 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 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.

RDAT2TAB

Function of RDAT2TAB

RDAT2TAB reads data provided in an RSM index (i.e. data) file. “RSM” is an acronym for “Regional Simulation Model”, this model having been developed by the South Florida Water Management District. Data contained within the data file is assumed to be real, though it makes no difference if, in fact, the data is comprised of integers. RDAT2TAB adds a coordinate to each data value, and writes data, coordinates, and element numbers to which they correspond to an output file in tabular format. Data contained in the latter file is thereby ready for display or contouring.

Using RDAT2TAB

RDAT2TAB asks the user for its input through a series of prompts. As for other members of the Groundwater Data Utility Suite, a single “e” (followed by <Enter>) supplied in response to any of these prompts will force RDAT2TAB to backtrack to the previous prompt.

RDAT2TAB issues only three prompts, these being as follows:-

```
Enter name of GMS two-dimensional mesh file:
Enter name of mesh element data file:
Enter name for tabular data output file:
```

See documentation of RSM2SRF for format of the two-dimensional mesh file. Note that this same format is also used by the GMS graphical user interface. An example of a mesh element data file (i.e. an “index file”) follows.

```
DATASET
OBJTYPE "network"
BEGSCL
ND 18
NAME "segment index"
TS 0 0.0
   3.45
   2.67
   7.65
   2.67
   8.54
   9.78
   2.13
   4.89
   10.32
   1.23
   5.98
   1.34
   1.56
   5.32
   1.89
   1.34
   8.42
   1.65
```

An index, or mesh element data, file employed by the RSM model.

Within this data file, each data item is associated with a mesh element. Data is supplied in order of increasing mesh element identifier. The latter are integers, and are supplied (not necessarily in increasing order) in the two-dimensional mesh file.

The tabular data file written by RDAT2TAB contains five columns. The first column contains mesh element identifiers, while the second and third contain mesh element centroid eastings and northings. Then follows mesh data as read from the mesh element data file. Finally the log of these data are presented in the fifth column. This can be useful if data represents, for example, hydraulic conductivities; it is often better to contour the log of these numbers than the numbers themselves.

Note that the order in which mesh elements are represented in the RDAT2TAB output file is the same as that in which they are represented in the two-dimensional mesh file. This may differ from that provided in the index file read by RDAT2TAB.

Uses of RDAT2TAB

Once RSM mesh data has been re-written by RDAT2TAB in tabular format, it can be contoured and/or posted onto maps of the model domain.

See Also

See also RSM2SRF, PPK2FACR and FAC2RSM.

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 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 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 such as QGIS) 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 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:
```

These files should both have the same filename base. They should have extensions of "*.mif*" and "*.mid*" respectively.

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.

When you import the *mif/mid* file pair into a GIS, it will ask you for the earth grid projection system that you are using. Presumably, the coordinates that are used in the model grid specification file use this same projection.

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 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 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 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.

RSM2SRF

Function of RSM2SRF

RSM2SRF reads a two-dimensional mesh file employed by the Regional Simulation Model (i.e. RSM model) developed by the South Florida Water Management District. This same file format is employed by the Groundwater Modelling System (GMS) graphical user interface developed by Brigham Young University. RSM2SRF then writes a number of files which can be read by SURFER for plotting of grid details. The following items can be plotted:-

1. node locations and numbers;
2. element centroid locations together with element numbers;
3. mesh element boundaries;
4. the total mesh boundary (which can also be employed for blanking purposes).

Using RSM2SRF

As for all members of the Groundwater Data Utility suite, if the user responds to any of RSM2SRF's prompts with an "e" (for "escape"), RSM will backtrack to its previous prompt. Thus recovery from user input error is immediate.

RSM2SRF commences execution by asking for the name of a two-dimensional mesh file. Its prompt is:-

```
Enter name of GMS two-dimensional mesh file:
```

in response to which the name of such a file must be provided. An example of a two-dimensional mesh file follows.

```
MESH2D
E3T     1    1     6     2     1
E3T     2    2     7     3     1
E3T     3    3     8     4     1
E3T     4    5    10     6     1
E3T     5    6    11     7     1
E3T     6    7    12     8     1
E3T     7    9    14    10     1
E3T     8   10    15    11     1
E3T     9   11    16    12     1
E3T    10    1     5     6     1
E3T    11    2     6     7     1
E3T    12    3     7     8     1
E3T    13    5     9    10     1
E3T    14    6    10    11     1
E3T    15    7    11    12     1
E3T    16    9    13    14     1
E3T    17   10    14    15     1
E3T    18   11    15    16     1
ND  1       0.000      15000.000 0.
ND  2    5000.000      15000.000 0.
ND  3   10000.000      15000.000 0.
ND  4   15000.000      15000.000 0.
ND  5       0.000      10000.000 0.
ND  6    5000.000      10000.000 0.
ND  7   10000.000      10000.000 0.
ND  8   15000.000      10000.000 0.
ND  9       0.000       5000.000 0.
ND 10    5000.000       5000.000 0.
ND 11   10000.000       5000.000 0.
ND 12   15000.000       5000.000 0.
ND 13       0.000          0.000 0.
ND 14    5000.000          0.000 0.
ND 15   10000.000          0.000 0.
ND 16   15000.000          0.000 0.
```

Example of a two-dimensional mesh file.

RSM2SRF next issues the following two prompts:-

```
Enter name for node coordinates file (<Enter> if none):
Enter name for centroid coordinates file (<Enter> if none):
```

If a filename is supplied in response to either of these prompts, RSM2SRF writes a three-column data file. Eastings and northings are recorded in the first two columns, while the last column contains integers; these integers are node numbers in the former case and element numbers in the latter case.

Next RSM2SRF asks the user if he/she would like two BLN files; these are plotted by SURFER as “base maps”. The prompts are:-

```
Enter name for mesh BLN file (<Enter> if none):
Enter name for mesh boundary BLN file (<Enter> if none):
```

In the former case every element of the mesh is represented. In the latter case only the outside of the mesh is represented. The latter can be employed for both display and contour blanking purposes.

Uses of RSM2SRF

As is apparent from its specifications, RSM2SRF allows RSM mesh geometric data to be re-written in a form where it can be imported and displayed by SURFER.

See Also

See also RDAT2TAB, PPK2FACR and FAC2RSM.

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 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.

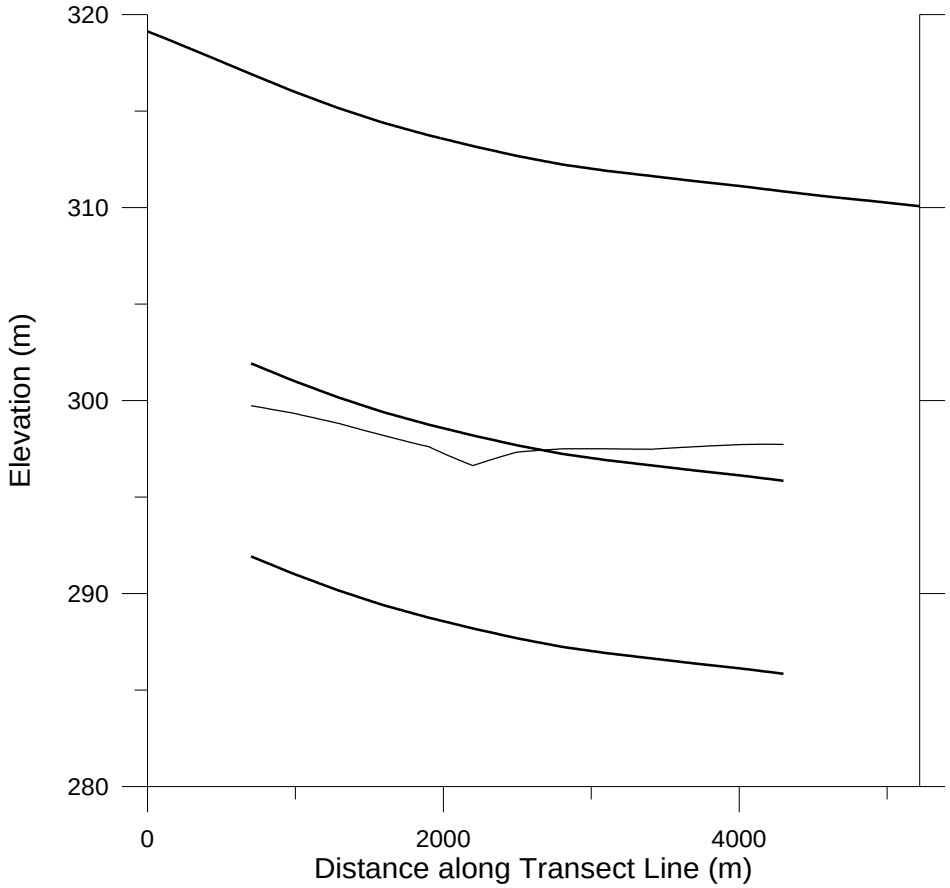
```
distance, easting,     northing,    surface.dat,  heads.dat
1060.000, 5.604295934, -664.798568, 31.45738,     26.45738
1070.000, 15.09112891, -661.636290, 31.54834,     26.45786
1080.000, 24.57796189, -658.474013, 31.67542,     26.45834
1090.000, 34.06479488, -655.311735, 31.84593,     26.45882
1100.000, 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.

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 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]:
```

and finally:

```
Enter number to add to elapsed time column (<Enter> if none):
```

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.) If a non-blank entry was supplied in response to SMP2HYD's final prompt, all entries in this column are incremented or decremented by the user supplied number. (This can be handy where time units of years are listed in this column; if, for example, the reference time is 1/1/1997, then 1997 can be added to all entries of this column to facilitate plotting of this data in graphing packages that do not handle dates well.) 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 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 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 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. 6.94×10^{-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 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.

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 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 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 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 5th column a bore sample file signifies dubious data in the 4th column. Such lines are ignored by SMPCAL, being used neither for data calibration (if occurring in the standard data file) nor for data adjustment (if occurring in the logger data file).

Next SMPCAL prompts:

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

All standard samples lying within a time period beginning *h* hours before the first logged sample for a particular 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.

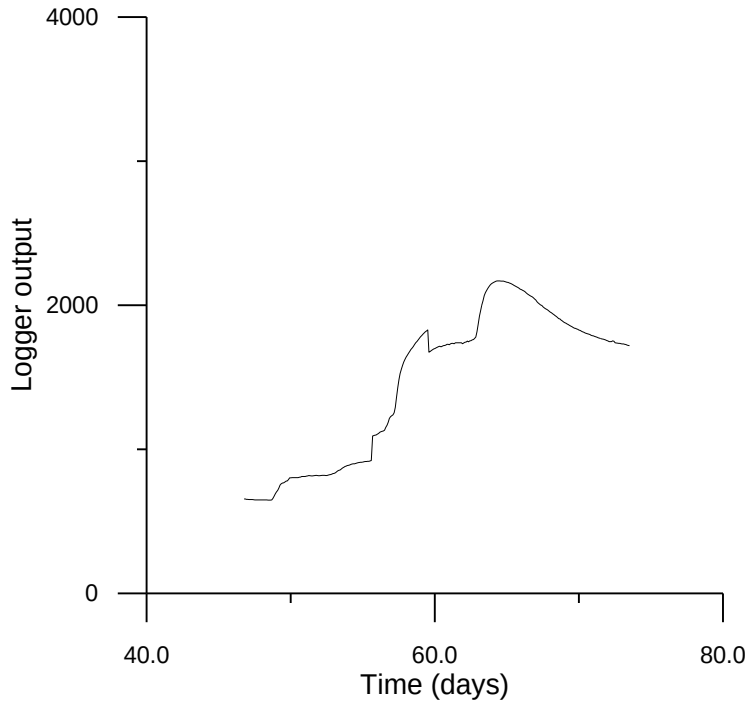
It is a general principal of data calibration that standard samples should span as wide a data range as possible. As is documented below, if this procedure is not followed data adjustment using SMPCAL can lead to unpredictable results. If neighbouring standard points have identical value, the calibration 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

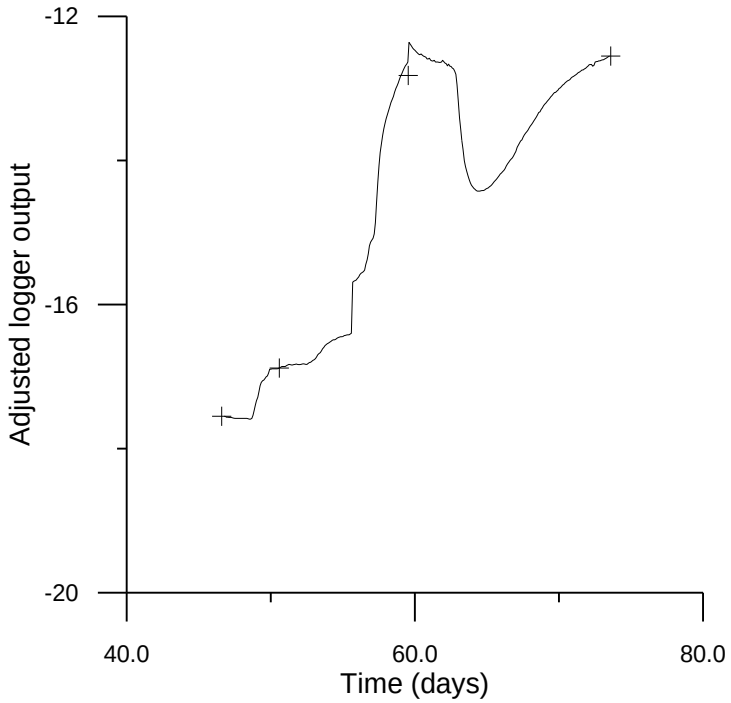
The most obvious use of SMPCAL is to calibrate logger data against manual readings taken over the time that the logger was operating. The fact that *m* and *c* (as determined

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.

Using 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 20 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.

SMPDIFF

Function of SMPDIFF

SMPDIFF reads an existing bore sample file. It writes another bore sample file in which sample values are replaced by sample differences. Differences can be taken between successive samples, between each sample and the first sample for each bore, or between each sample and a reference level. Any of these can be useful in formulation of the objective function used in calibration of a transient groundwater model.

Using SMPDIFF

As for most members of the Groundwater Data Utility suite, SMPDIFF commences execution by reading a file named *settings.fig* in which date and MODFLOW array conventions are listed. If this file is not present in the directory from which SMPDIFF is run, or if this file has errors, SMPDIFF ceases execution with an error message.

Next SMPDIFF issues a series of prompts. By responding with "e" (followed by <Enter>) to any one of these prompts, execution can be returned to the previous prompt.

First SMPDIFF asks:-

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

in response to which the name of a bore sample file should be provided (this file type is referred to as a "site sample file" in PEST Surface Water Utility documentation). It then prompts for the name of a bore listing file:-

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

As is described elsewhere in this documentation, a bore listing file provides a list of bores, one to a line. Only bores cited in this file will be processed by SMPDIFF, and hence will appear in the SMPDIFF output file; other bores listed in the original bore sample file will be ignored.

Next SMPDIFF asks:-

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

as well as for the specifics on how sample differences are to be computed:-

```
Difference wrt previous, first or reference sample?  [p/f/r]:
```

If the response is "p", then the new bore sample file lists differences between each sample and its previous sample; the difference is attributed to the time of the latter sample. If the response is "f", then the sample pertaining to the first occurrence of each bore is subtracted from that pertaining to all later occurrences. If the response is "f" then a bore-specific reference value is substracted from each sample. In this case SMPDIFF asks:-

```
Enter name of reference value file:
```

An example of a reference value file follows:-

```
rb_3432      35.432
rb_453       72.85
rb_4235      85.32
etc
```

Part of reference value file.

Each line of a reference value file must have two entries. The first is the name of a bore while the second is a reference value for that bore. This is the value subtracted from all bore sample values if "r" is supplied for the above prompt.

Finally, if differences are to be taken between subsequent samples, or between the first sample associated with a bore and each later sample for that bore, SMPDIFF asks:-

```
Retain first of each sample in output file?  [y/n]:
```

If the responds to this prompt is "y" the first sample in the bore sample file of each bore cited in the bore listing file (these being the bores for which sample differencing is undertaken) will be transferred directly to the new bore sample file. No differencing is undertaken for this sample (the difference would be zero anyway). Thus its undifferenced value is retained.

Uses of SMPDIFF

SMPDIFF is useful in calibration of a transient model. In this capacity it is run on both an observation bore sample file and on its model-generated equivalent as produced by MOD2OBS. It allows temporal head differences and/or drawdowns to be directly included in the inversion process. This is an almost mandatory part of the transient model calibration process, for it is more important that model outputs reproduce temporal variations of water levels, then the water levels themselves if, in fitting the latter hydraulic properties are compromised in order to make up for model structural inadequacies.

See Also

See also SMPTREND, SMP2SMP and PESTPREP.

SMPSTAT

Function of SMPSTAT

SMPSTAT reads a bore sample file (also known as a site sample file). For each bore that it finds in the file, it calculates a number of statistics pertaining to the time series that is associated with that bore. These statistics are recorded in a user-nominated, text output file.

Using SMPSTAT

SMPSTAT has only two prompts. These are for the name of the bore sample file that it must read, and the name of the text file that it must write. The following figure shows part of a SMPSTAT output file.

```
Site: sitea ----->
   Number of samples                  : 28
   First sample at                    : 01/01/1986  12:00:00
   Last sample at                     : 30/12/1992  12:00:00
   Days spanned by time series        : 2555.0000
   Mean of values                     : 103.5000000000
   Standard deviation of values       : 2.449489742783
   Time-average of values             : 103.5000000000
   Minimum value                      : 100.0000000000
   Maximum value                      : 107.0000000000

Site: siteb ----->
   Number of samples                  : 3
   First sample at                    : 30/07/1993  12:00:00
   Last sample at                     : 30/07/1994  00:00:00
   Days spanned by time series        : 364.5000
   Mean of values                     : 3037.333333333
   Standard deviation of values       : 0.5773502681142
   Time-average of values             : 3037.498628258
   Minimum value                      : 3037.000000000
   Maximum value                      : 3038.000000000
```

Part of a file written by SMPSTAT

Uses of SMPSTAT

As presently programmed, the following statistics are recorded for each bore encountered in the bore sample file which SMPSTAT reads:

- The number of data values associated with that bore;
- The date and time of the first and last data items;
- The number of days spanned by this time interval (fractional if necessary);

- The mean and standard deviation of data values;
- The maximum and minimum data values;
- The time-average of data values.

The time average of data values is obtained by integrating data values with respect to time (using the trapezoidal rule) and then dividing by the time span of these values.

See Also

See also other programs whose names begin with “SMP”.

SMPTREND

Function of SMPTREND

Like SMPDIFF, SMPTREND was written to assist in calibration of a transient model. Like SMPDIFF, SMPTREND creates a new bore sample file based on values in an existing bore sample file. And like SMPDIFF these new values are computed by differencing. However sampling is restricted to a sample window within each year. Thus SMPTREND allows the calibration process to focus in variations that exist over many seasons rather than just over a single season, for by restricting sampling to only a certain window of the year, within-year variation is not recorded in the SMPTREND-written bore sample file. Also, because differences are taken, the calibration process is able to focus on variations rather than absolutes, and thus has the ability to supply values to these parameters which govern these long-term variations.

Using SMPTREND

As for most members of the Groundwater Data Utility suite, SMPTREND commences execution by reading a file named *settings.fig* in which date and MODFLOW array conventions are listed. If this file is not present in the directory from which SMPTREND is run, or if this file has errors, SMPTREND ceases execution with an error message.

Next SMPTREND issues a series of prompts. By responding with “e” (followed by <Enter>) to any one of these prompts, execution can be returned to the previous prompt.

First SMPTREND asks:-

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

in response to which the name of a bore sample file should be provided. It then prompts for the name of a bore listing file:-

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

As is described elsewhere in this manual, a bore listing file provides a list of bores, one to a line. Only bores cited in this file will be processed by SMPTREND, and hence will appear in the SMPTREND output file; other bores listed in the original bore sample file will be ignored.

Next SMPTREND asks:-

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

This is the name of the bore sample file which it will write.

SMPTREND’s next prompts are:-

```
Enter opening day of year for trend sampling [days]:
Enter closing day of year for trend sampling [days]:
```

Collectively the responses to these two prompts define the sampling window in which SMPTREND operates. So if, for example, the user responds with "60" to the first of the above two prompts and with "90" to the second of the above two prompts, SMPTREND will ignore any samples within its input bore sample file that pertain to any day of the year (starting from January 1) prior to the 60th day or after the 90th day. Furthermore, only the first sample within that time interval in any one year is read from the input file and processed for writing to its output file.

SMPTREND's next prompt is:-

```
Difference wrt to first or reference sample?  [f/r]:
```

If the response to this prompt is "f", then SMPTREND will write a bore sample file in which the difference is taken between the first sample in the sample window for each year, and the first sample in the sample window ever encountered in the bore sample file. If the response is "r" then a bore-specific reference value is subtracted from the first sample in the sample window encountered in each year. These references values are read from a reference value file, the name of which must be supplied in response to the following prompt:-

```
Enter name of reference value file:
```

The format of a reference value file is provided in documentation for SMPDIFF.

Finally, if differencing with respect to the first encountered sample within the sample window was requested, SMPTREND prompts:-

```
Retain initial trend sample for each bore in output file?  [y/n]:
```

If the answer to this question is "y", then the first sample is passed straight through to the output file (with no difference taken with itself). Otherwise a sample pertaining to this initial sample time is not recorded on the SMPTREND output file. Note that if differencing is performed with respect to reference samples, the first sample for any bore encountered in the sample window is indeed differenced with the pertinent reference value, and the difference recorded on the SMPTREND output bore sample file.

Uses of SMPTREND

In calibration of a model, it is often a good idea to isolate different facets of the response of a system into different components of the objective function by assigning them to different observation groups. Observations can then be weighted such that the contribution to the objective function by each group is about the same; thus information in the calibration dataset that is most informative of one data type is not "drowned out" by information pertaining to another data type.

In calibration of a long-term transient groundwater model, seasonal water table fluctuations are often very informative of both specific yield, and of the parameters pertaining to an implicit or explicit recharge submodel used by the groundwater model. Longer term variations are often more informative of the latter. SMPTREND provides a means of isolating within-year seasonal variations from long-term trend variations. Furthermore, the differencing process allows the calibration process to

adjust parameters so that changes in heads rather than absolute heads are fit, this often resulting in more accurate estimation of the parameters governing water level changes, as parameters that govern absolute water levels are not adjusted to accommodate model inadequacies that may be responsible for long-term bias in model outputs at some sites. Thus, a model's inability to match heads in a steady state context need not compromise its ability to match head variations.

SMPTREND should be used to process observed head data, and the model-generated equivalents of observed head data as written by MOD2OBS; in this latter capacity it should be run as part of the model.

See Also

See also SMPDIFF, PESTPREP, SMP2SMP, MOD2OBS.

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 1.70141×10^{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 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 Part A 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 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 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 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 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 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 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 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 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 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.