Groundwater Data Utilities

Part A: Overview

Watermark Numerical Computing

August, 2021

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).

- The development of utilities which support the interface between PEST and the MODFLOW-2005 adjoint process developed by Tom Clemo was supported by a contract with Boise State University under EPA Grant X-96004601-0. So too were utilities supporting the use of calibration-constrained Monte-Carlo as a tool for model predictive uncertainty analysis.

- 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 (particularly those associated with unstructured grid models) were written with support from the National Centre for Groundwater Research and Training, Flinders University, Australia. The help of the Centre is greatly appreciated.

- Support for the writing of unstructured grid utilities was also provided by the Queensland Hydrology Unit, Science Delivery Division, Department of Science, Information Technology, Innovation and the Arts (DISITIA), and by the Office of Groundwater Impact Assessment, Queensland Government Department of Natural Resources and Mines.

- Support for the writing of MODLFOW 6 utilities was provided by the GMDSI project funded by BHP and Rio Tinto.

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

Disclaimer

Support

To obtain support in the use of the Groundwater Data Utilities or to report any bugs, please contact:

Dr. John Doherty
Watermark Numerical Computing
Australia
Email: johndoherty@ozemail.com.au

Table of Contents

See the prefaces to Parts B and C of this manual for tables of the Groundwater Data Utilities which include a brief description of the role of each.

The utilities that are documented in Part C of this manual were developed specifically for use with unstructured grid models, particularly MODFLOW-USG and MODFLOW 6.

See also documentation of the following programs. These have their own manuals.

- **PLPROC, a parameter list processor written with pilot points and unstructured grids in mind;**

- **OLPROC, a model observation processor and PEST input dataset constructor;**

- **TS6PROC, a program which allows adjustable parameterization of MODFLOW 6 time series.**

1. Introduction

1.1 General

In spite of the availability of powerful, user-friendly, model-specific pre- and postprocessing software, groundwater modelling is difficult. One of the main reasons for this is the necessity to transfer large amounts of data into and out of models, and between models, databases and visualisation and display packages. Many an attempt to model an important groundwater system has died a “death by a thousand cuts” as the difficulties of model-related data compilation and processing tap the time, patience and resources of the groundwater modeller.

The software documented in this manual, written to complement existing models and their graphical pre- and postprocessors, was developed in an attempt to overcome some of the ancillary problems attached to the practice of groundwater modelling. None of the tasks performed by any of the programs documented herein is particularly difficult; yet to undertake these tasks without software assistance would be time-consuming and error-prone, as they generally involve the processing of large amounts of data.

It is hoped that the Groundwater Data Utilities documented in this report will free modellers of some of the more laborious data-handling tasks which accompany all modelling projects, so that they may then be free to turn greater attention to the critically important issues of correct model setup, parameterisation and calibration. If this does, indeed, occur then the programming and documentation of the Groundwater Data Utilities described in this manual will have been worth the considerable effort required for their development.

1.2 The Software Environment

Many of the utilities documented in this report have been written specifically for use with the MODFLOW family of models (including MODFLOW-USG and MODFLOW 6), as well as closely-related programs such as MT3D and SEAWAT. With little or no modification these utilities can also be used with other cell-centred finite-difference models. Others of the utilities are model-independent, being used simply for groundwater data manipulation and presentation, whether such processing is required as part of a model-construction exercise or not.

The Groundwater Data Utilities can be used in conjunction with any MODFLOW graphical user interface (GUI). However it is important that the GUI allow the importing and exporting of MODFLOW-compatible data arrays. Where any of the utilities documented herein read or write two-dimensional, real or integer arrays for structured versions of MODFLOW (and MT3D), two slightly different conventions for formatted array storage are adopted. The first convention requires that a header line recording the number of model columns and rows precedes the array itself in its formatted file; the second convention requires that this header line be omitted. The ability to use either of these conventions ensures that programs of the Groundwater Data Utilities can be used in conjunction with any commercial MODFLOW GUI. The user selects the convention that he/she wishes to use in his/her work through an appropriate entry in a “settings file”.

Where required by any of the utilities, grid geographical information is supplied through a “grid specification file”. For structured grids, this file can be easily prepared by the user based on information available through any MODFLOW GUI, and from the MODFLOW input files produced by that GUI.

Some of the Groundwater Data Utilities allow model-generated data to be interpolated and/or reformatted for importation into graphing and contouring software. Some of these utilities are independent of the user’s choice of graphing or contouring software; others, however, relate specifically to Golden Software’s SURFER package. Some of the latter utilities provide a mechanism whereby SURFER can be used in a model preprocessing role, while others enhance its ability to display model results with or without diagrammatical representation of the finite difference grid (including any zonation defined within this grid).

Functionality is included within the Groundwater Data Utilities for data exchange between a model and a Geographical Information System (i.e. GIS). In particular, model geographical, real and integer array data can be translated to MAPINFO mif/mid format. The contents of model real and integer arrays can be read as tables, allowing GIS-exported real and integer array data to be incorporated into a model.

An important role performed by many of the Groundwater Data Utilities is the provision of assistance in model parameter definition, and in calculating the model-generated equivalents to field observations of head and flow. These, in conjunction with other utilities which automate the construction of input files for PEST, facilitate the use of nonlinear parameter estimation techniques in the calibration of groundwater models.

None of the Groundwater Data Utilities are able to read database and spreadsheet files. However many of them have been written to extract or process data of the type normally held in these types of file. To save programming effort and maintain independence from any particular database or platform, the utilities have been written to interface with “flat” ASCII files of a type that can be easily exported from a user’s groundwater database or spreadsheet. In most cases, creation of a file type which adheres to the conventions used by the Groundwater Data Utilities can be achieved with little difficulty.

1.3 Features in Common

Though each of the programs comprising the Groundwater Data Utilities performs its own specific aspect of groundwater model pre- and/or postprocessing, all of the utilities share some common characteristics. Each requires that the user supply information in response to command-line prompts. While this can be a cumbersome method of communication between a program and its user, inconvenience has been mitigated somewhat by the fact that a user can always "backtrack" in program execution by entering "`e`" (for "escape") at any prompt. Thus, whether a text string or number is expected, a simple "`e`" will cause the program to display its previous prompt; responding to this prompt with another "`e`" will make the program display the prompt before that, etc. Hence if, in the process of replying to a succession of screen prompts, incorrect data is entered at any stage, a user can "wind the program back" to the point at which previous data entries were all correct, and recommence execution from that point.

Most of the Groundwater Data Utilities read input files. Most perform exhaustive checking of these input files as they are read, documenting the line numbers and types of errors (if any) contained therein.

When undertaking model pre- and postprocessing using the Groundwater Data Utilities, certain files need to be read again and again. To ease the tedium of having to provide the names of one or more of these files each time a utility executes, their names can be placed into a "filename file" residing in the directory from which the utilities are run (i.e. the "current directory"); this file must be named `files.fig` (note lower case in UNIX environment). Each utility tries to read `files.fig` from the current directory as soon as it commences execution. If it finds `files.fig` and reads it without error, the utility provides the user with default filenames (read from `files.fig`) when prompting him/her for the names of certain files. In each case the user can then either accept the default (as read from `files.fig`) or enter an alternative filename. Figure 1.1 shows a filename file.

```
grid_specification_file=ex.spc
bore_coordinates_file=bores.crd
bore_sample_file=elev2.dat
bore_pumping_file=pump.dat
pilot_points_file=points.dat
node_to_bore_file=n2binterp.dat
```

Figure 1.1 A filename file.

A filename file can contain up to six lines of data, each line containing two items. The first is the name of a particular file type (with an underscore substituted for the space between words) while the second identifies the pertinent file. Note that full paths can be included in filenames. Note also that filenames containing spaces should be surrounded by quotes.

At present, only five types of file can be included in a filename file; these are easily identified in Figure 1.1. These file types were chosen for inclusion as there is often only one file of each of these types required over the entirety of a modelling exercise. Other commonly used file types (such as the bore listing file) were not chosen for inclusion in the filename file as different files of these types are likely to be required depending on the type of pre- or postprocessing task that is being undertaken.

A particular type of file that must reside in the directory from which the Groundwater Data Utilities are run is a "settings file". A settings file must be named `settings.fig` (note lower case in the UNIX environment). **Unlike the filename**

file whose presence is not essential, it is obligatory for a settings file to be present if certain of the utilities are to run; if any of these utilities are invoked when a settings file is not present, the respective program will terminate execution with an appropriate error message. The utilities that require the presence of a settings file are those that read and/or write date and time information and those that read and/or write MODFLOW-compatible real and integer arrays. An example of a settings file is shown in Figure 1.2

```
date=dd/mm/yyyy
colrow=yes
```

Figure 1.2 A settings file.

One of the two possible lines contained in a settings file (the first in Figure 1.2), informs the utilities of the date protocol to use in all input/output files, and in keyboard and screen interaction with the programs. In Figure 1.2, the day precedes the month; the alternative is "mm/dd/yyyy", in which case the month precedes the day. The other entry informs the utilities of the convention to use for formatted MODFLOW-compatible real and integer array storage. If `colrow` is set to "yes", those utilities which read such arrays expect the array to be preceded by a header containing two numbers, viz. the number of columns and rows (in that order) comprising the finite difference grid. If `colrow` is set to "no", such a header is not expected. Those utilities which generate MODFLOW-compatible real and integer arrays observe the same protocol when writing these arrays to a file.

1.4 Unformatted Files

Unfortunately, unformatted files generated by programs compiled by different FORTRAN compilers are not always interchangeable. It is thus possible that those of the utilities which read unformatted MODFLOW and/or MT3D output will not be able to read files generated by versions of these models compiled by certain FORTRAN compilers. If this occurs, contact me and I will send you a version of these programs that will read files generated by your particular model.

1.5 This Manual

This manual is divided into three parts. Part A, this part, provides an overview of the Groundwater Data Utilities and discusses aspects common to all of them. Parts B and C document each utility individually. (Utilities written specifically for unstructured grid models are documented in Part C.)

See also the manuals for PLPROC, OLPROC and TS6PROC. Pilot point interpolation functionality provided by PLPROC supersedes that provided by members of the Groundwater Data Utilities. Functionality provided by OLPROC automates PEST input dataset construction in certain modelling contexts, particularly those associated with use of MODFLOW 6.

2. File Types

2.1 Introduction

This section lists the types of files used by the Groundwater Data Utilities. Most of the Utilities read one or more of, and/or generate one or more of, the types of files listed within this section.

Parts B and C of this manual describes the Groundwater Data Utilities on a program-by-program basis. Within the program descriptions of Parts B and C many references are made to the file types documented below.

2.2 Bore Coordinates File

A bore coordinates file lists bore identifiers, bore coordinates and, optionally, model layer numbers associated with a collection of bores (i.e. wells). Figure 2.1 shows part of a bore coordinates file.

```
13500002A    432757.001     7251364.668 1
13500002B    432532.650     7251332.776 2
13500005A    431938.751     7252252.603 1
13500006A    432189.994     7252530.667 1
13500007A    431794.596     7253020.996 1
13500008A    431852.452     7252682.867 1
13500009A    431992.097     7252806.516 1
13500012A    430844.466     7252185.675 1
13500015A    431043.509     7251663.817 1
13500017A    432970.533     7253641.892 1
13500023A    432885.506     7253795.232 1
13500032A    430646.701     7252430.711 1
13500032B    430700.251     7252953.946 2
13500032C    431519.999     7251789.264 3
13500042A    432406.371     7254192.785 1
13500047A    432185.152     7253514.981 1
13500049A    432018.367     7253175.856 1
13500050A    432184.395     7253668.732 1
13500052A    427669.234     7258198.290 1
13500053A    427556.282     7258320.658 1
13500054A    426825.900     7258378.399 1
13500055A    426378.631     7258006.921 1
13500056A    424855.152     7259321.369 1
```

Figure 2.1 Extract from a bore coordinates file.

Each line of a bore coordinates file should possess at least three items. The first item is a bore identifier; this should consist of a maximum of 20 characters (shorter is recommended). For the Groundwater Data Utilities the bore identifier is case insensitive, being converted internally to upper case by each utility.

The second and third items on each line of a bore coordinates file consist of the east and north (i.e. *x* and *y*) coordinates of the bore whose identifier comprises the first item on the line. The optional fourth item should be an integer indicating the model layer number to which the bore pertains. Many of the Groundwater Data Utilities do not require that the fourth (layer number) column be present in a bore coordinates file. However for those programs that require a bore's layer number, the fourth column is essential.

Bores can be listed in any order in a bore coordinates file. However a bore coordinates file must not reference the same bore twice.

2.3 Bore Information File

A bore information file supplies information about one or more bores. A bore coordinates file is one specific instance of a bore information file. Figure 2.2 shows part of another bore information file.

```
  13500002A 23.42  1.43e-4
  13500002B 34.34  2.43e-3
  13500005A 27.45  3.00e-3
  13500006A 15.34  1.09e-2
  13500007A 19.45  9.43e-3
  13500008A 23.43  6.43e-3
  13500009A 14.56  9.32e-4
  13500012A 15.76  1.43e-2
  13500015A 39.43  1.02e-2
  13500017A 33.76  7.53e-3
  13500023A 32.87  6.32e-3
  13500032A 29.02  1.34e-2
  13500032B 31.43  9.93e-4
  13500032C 27.67  7.65e-3
  13500042A 28.29  6.34e-3
  13500047A 23.67  2.39e-3
  13500049A 41.83  1.43e-4
  13500050A 34.98  2.43e-3
  13500052A 35.84  2.84e-3
  13500053A 28.93  7.43e-5
  13500054A 37.65  8.87e-3
  13500055A 20.59  3.94e-3
```

Figure 2.2 Extract from a bore information file.

Each line of a bore information file should possess at least two items. The first item is a bore identifier, an ASCII string of 20 characters or less in length. For the Groundwater Data Utilities the bore identifier is case-insensitive, all identifiers being converted to upper case internally.

Subsequent items on each line of a bore information file record different types of information pertaining to each bore. In Figure 2.2 the second column records measured water elevations while the third column contains pollutant concentrations; all measurements pertain to samples gathered at the same time (or nearly the same time), thus constituting a "snapshot" of aquifer conditions. Items within each line may be separated by whitespace (including tabs) or commas. Each line should contain the same number of items in order to maintain column consistency throughout the file.

It is important to note that a bore information file should not cite any bore more than once. Hence bore pumping files (Section 2.5) and bore sample files (Section 2.6) are not bore information files.

2.4 Bore Listing File

A bore listing file simply provides a list of bores, one to a line. Figure 2.3 shows part of such a file.

```
 13500002A
 13500002B
 13500005A
 13500006A
 13500007A
 13500008A
 13500009A
 13500012A
 13500015A
 13500017A
 13500023A
 13500032A
 13500032B
 13500032C
 13500042A
 13500047A
 13500049A
 13500050A
 13500052A
 13500053A
 13500054A
 13500055A
```

Figure 2.3 Extract from a bore listing file.

Different bore listing files are often used in conjunction with a single bore coordinates file, thus providing a mechanism whereby a subset of the latter can be selected for a particular type of groundwater data processing. Thus whenever a program reads both a bore coordinates file and a bore listing file, an error condition will arise if bores listed in the latter file are not also cited in the former file. However the reverse is not the case.

A bore listing file may possess more than one data column. If it does, only the first item on each line is read. Thus a file supplied to a program as a bore coordinates file can also be supplied to the same program as a bore listing file if it is desired that all bores in the former file be subject to processing of the type carried out by that utility.

For the Groundwater Data Utilities documented in this manual, bore identifiers are case insensitive, being translated internally to upper case.

No bore should be cited more than once in a bore listing file. If it is cited more than once, an error condition will arise and an appropriate error message will be displayed.

2.5 Bore Pumping File

A bore pumping file records the history of borehole extraction within a study area. Figure 2.4 shows a fragment of such a file.

```
 40050    20/06/1989 12:00:00   18/09/1989 12:00:00  0.000
 40050    18/09/1989 12:00:00   18/12/1989 12:00:00  13.00
 40050    18/12/1989 12:00:00   20/03/1990 12:00:00  65.00
 40050    20/03/1990 12:00:00   27/06/1990 12:00:00  2.000
 40050    27/06/1990 12:00:00   25/09/1990 12:00:00  3.000
 40050    25/09/1990 12:00:00   20/12/1990 12:00:00  41.00
 40050    20/12/1990 12:00:00   26/03/1991 12:00:00  58.00
 40050    26/03/1991 12:00:00   25/06/1991 12:00:00  34.00
 40050    25/06/1991 12:00:00   29/09/1991 12:00:00  25.00
 40050    29/09/1991 12:00:00   09/01/1992 12:00:00  25.00
 40050    09/01/1992 12:00:00   12/03/1992 12:00:00  35.00
 40050    12/03/1992 12:00:00   26/06/1992 12:00:00  6.000
 40050    26/06/1992 12:00:00   14/09/1992 12:00:00  0.000
 40050    14/09/1992 12:00:00   04/01/1993 12:00:00  23.00
 40051    04/01/1988 12:00:00   18/03/1988 12:00:00  13.00
 40051    18/03/1988 12:00:00   15/06/1988 12:00:00  0.000
 40051    15/06/1988 12:00:00   16/09/1988 12:00:00  2.000
 40051    16/09/1988 12:00:00   28/10/1988 12:00:00  0.000
 40051    28/10/1988 12:00:00   22/11/1988 12:00:00  3.000
 40051    22/11/1988 12:00:00   04/01/1989 12:00:00  0.000
 40051    04/01/1989 12:00:00   25/01/1989 12:00:00  0.000
 40051    25/01/1989 12:00:00   21/02/1989 12:00:00  10.00
```

Figure 2.4 Extract from a bore pumping file.

Each line of a bore pumping file must contain 6 entries. The first entry is the bore identifier which must be of 20 characters or less in length. When read by programs of the Groundwater Data Utilities the bore identifier is case-insensitive, being converted internally to upper case. (Note that coordinates corresponding to these bores will generally reside in an accompanying bore coordinates file; see Section 2.2 of this manual.) The next two items on each line of a bore pumping file define the starting date and time of a pumping interval. Depending on the contents of the settings file `settings.fig` (see Section 2.14), the date must be presented either in the format `dd/mm/yyyy` or `mm/dd/yyyy;` time must be recorded in the format `hh:mm:ss.` The next two items denote the end of a pumping interval using the same notation. The final item on each line lists the amount of pumping that has taken place during the nominated time interval; units are arbitrary.

The following rules must be observed when generating a bore pumping file:

- On any line of a bore pumping file the first date/time must precede the second date/time (i.e. the first date must precede the second date or, if the dates are the same, the first time must precede the second time).
- For the same bore, the first date and time on any given line of a bore pumping file must be the same as the second date and time on the previous line.
- All entries pertaining to any given bore must follow one after the other (in temporal sequence, as noted above); a sequence of entries for a certain bore must not be interspersed with a sequence of entries for another bore.

In most cases a bore pumping file will have been either downloaded from a user’s groundwater database or constructed from a file downloaded from such a database. Where a modeller is working in an area of high groundwater usage, the bore pumping file may be very large. While this poses no problem to members of the Groundwater Data Utilities suite, there is a software-imposed limit to the number of entries permitted for a single bore. In all cases this limit is easily adjusted by making a minor source code alteration. The nature of this alteration will be provided as an appropriate error message in the unlikely event of its being required.

Both positive and negative pumping figures are permitted in the final column of the bore pumping file.

2.6 Bore Sample File

A bore sample file records data gathered at discrete sample times at a number of specific locations, e.g. water level or chemical concentration data gathered through borehole sampling programs. Each line of a bore sample file has four, possibly five, entries, each of which must be separated from its neighbouring entry by one or more whitespace (including tab) characters. Typically a bore sample file will hold data extracted from a groundwater database. There is no limit to the size of this file as all Groundwater Data Utility programs which read it hold only part of its contents in memory at any one time. However there is a limit to the maximum number of entries allowed for any one bore. This limit is easily adjusted by making a slight alteration to pertinent source code files. The nature of this adjustment will be provided as part of an appropriate error message in the unlikely event of its being required. Figure 2.5 shows part of a bore sample file.

```
13500002A    25/09/1991   12:00:00  12.00
13500002A    02/01/1992   12:00:00  11.83
13500002A    24/03/1992   12:00:00  12.81
13500002A    29/06/1992   12:00:00  13.54
13500002A    22/09/1992   12:00:00  13.24
13500002A    17/12/1992   12:00:00  12.84
13500002A    22/03/1993   12:00:00  12.38 x
13500002A    21/06/1993   12:00:00  11.83 x
13500002A    27/09/1993   12:00:00  11.61 x
13500002A    16/12/1993   12:00:00  12.35
13500002A    01/03/1994   12:00:00  11.79
13500002A    22/03/1994   12:00:00  11.89
13500005A    19/02/1959   12:00:00  29.84
13500005A    05/03/1959   12:00:00  30.33
13500005A    20/03/1959   12:00:00  30.76
13500005A    06/04/1959   12:00:00  31.19
13500005A    17/04/1959   12:00:00  31.45
13500005A    01/05/1959   12:00:00  31.65
13500005A    15/05/1959   12:00:00  31.65
13500005A    29/05/1959   12:00:00  31.65
13500005A    12/06/1959   12:00:00  31.65
13500005A    26/06/1959   12:00:00  31.46
13500005A    10/07/1959   12:00:00  31.34
```

Figure 2.5 Extract from a bore sample file.

The first item on each line of a bore sample file is a bore identifier. This identifier must be of 20 characters or less in length. When used with programs of the Groundwater Data Utilities the bore identifier is case-insensitive. The second item is the date; depending on the contents of the settings file `settings.fig`, this must be expressed either in the format `dd/mm/yyyy` or `mm/dd/yyyy`. Then follow the time (in the format `hh:mm:ss`) and the borehole measurement pertaining to the cited date and time. An optional fifth item may be present on any line; if present this item must consist solely of the single character "`x`" to indicate that the previous data element lacks integrity (for example if a borehole water elevation measurement must be treated with suspicion because the bore may have been dry when the measurement was taken).

The following rules must be observed when generating a bore sample file:

- For any one bore, dates and times must be listed in increasing order.
- All entries for the same bore must be in juxtaposition; in other words, it is not permitted to list some of the entries for a particular bore in one part of a bore sample file and the remainder of the entries in another part of the same file, with data pertaining to one or more other bores in between.

As most groundwater data is sampled at a measurement interval that far exceeds one day, measurement time-of-day is rarely recorded in the field. Thus a "notional" measurement time (for example midday as shown in Figure 2.5) can be attributed to all samples.

2.7 Filename File

A filename file (which must possess the name `files.fig`) contains the names of files which are likely to be repeatedly read by programs of the Groundwater Data Utilities. When a program requiring the name of such a file is run, the pertinent filename, as read from the filename file, is displayed in square brackets following the prompt. The user may then either press the <Enter> key to accept the default, or type in the correct filename him/herself.

As shown in Figure 2.6, only six types of file can currently be represented in a filename file. These can be listed in any order. References to any of these files can be omitted from the filename file if desired. Note also that if a filename contains a space, it should be enclosed in quotes.

```
grid_specification_file=ex.spc
bore_coordinates_file=bores.crd
bore_sample_file=elev2.dat
bore_pumping_file=pump.dat
pilot_points_file=points.dat
node_to_bore_file=n2binterp.dat
```

Figure 2.6 A filename file.

The file types listed in Figure 2.6 were selected for inclusion in the filename file because different members of the Groundwater Data Utilities are likely to repeatedly access these same files. Thus, for example, a bore coordinates file can list the coordinates of all bores found within a certain project area; different bores can be selected for different types of pre- and postprocessing using different bore listing files. Similar considerations apply to the bore sample file, the bore pumping file, and the node-to-bore interpolation file. The grid specification file is likely to be unique to a particular project area.

Use of a filename file is optional. If present it will only be read by a member of the Groundwater Data Utilities if it resides in the directory from which that utility was run. If absent, a Groundwater Data Utility will not display a default filename in brackets when prompting the user for the name of any of the types of file listed in Figure 2.6.

2.8 Grid Specification File

A grid specification file contains all of the information required to draw a structured finite difference grid on a map using real-world coordinates. Thus information contained in this file can be used to determine the positions of bores with respect to the grid, superimpose model outcomes on other geographical data, and much more. Many of the Groundwater Data Utilities begin execution by reading a grid specification file. Some of the utilities simply need to read the grid dimensions while others need to know the full complement of geographical information encompassed in the grid design. Figure 2.7 shows the grid specification file for a small grid consisting of 8 rows and 10 columns. Figure 2.8 shows the grid to which it pertains.

```
8 10
300000.0  7800000.0 30.0
2*100 6*50 2*100
50  50  50  50  50  50  50  50
```

Figure 2.7 A grid specification file.

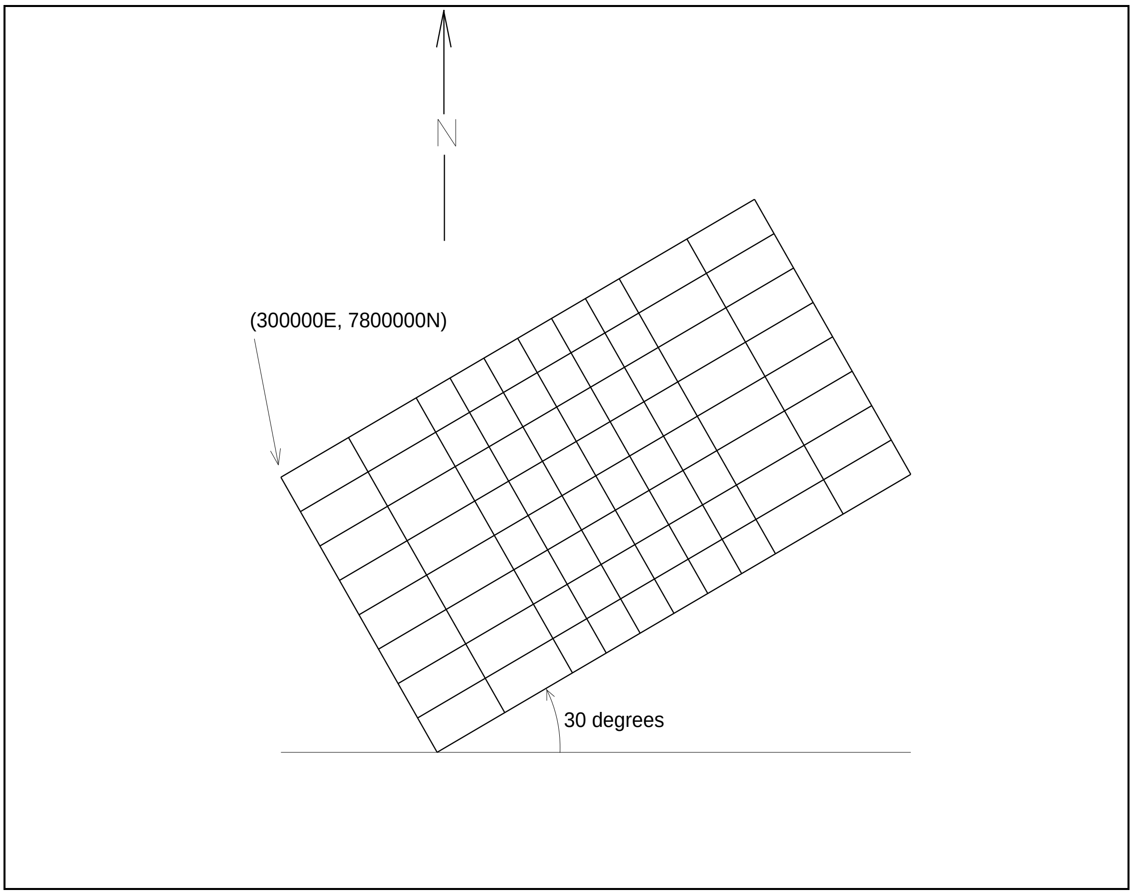

Figure 2.8 The finite difference grid whose grid specification file is shown in Figure 2.7

Each line of a grid specification file contains two or more items; each item must be separated from its neighbours by a space, comma or tab. The first line contains two items, viz. the number of rows and columns respectively, within the finite-difference grid. (It should be noted that, where it is used, the header to a formatted, MODFLOW-compatible, real or integer array contains these grid dimensions in reverse order, the result of an unfortunate historical accident. Use, or otherwise, of this header in formatted array storage is determined by the `colrow` setting in the “settings file”, `settings.fig`; see Section 2.14 for further details.)

The second line of a grid specification file is comprised of three entries. These are the east and north (ie. *x* and *y*) coordinates of the top left corner of the finite difference grid, and the rotation of the grid row direction with respect to east (positive for anticlockwise, negative for clockwise); see Figure 2.8. This angle must be expressed in degrees and must lie between -180 and 180 degrees.

The third line of a grid specification file holds the MODFLOW *delr* vector. This is a list of grid row widths in order of increasing column index. There should be as many entries in this vector as there are columns in the finite difference grid. Note that it is not necessary to write all elements of the *delr* vector on the one line if the line becomes too long. All programs which read a grid specification file will continue reading subsequent lines until enough elements have been read.

Next, on a new line, follows the MODFLOW *delc* vector, i.e. a list of grid cell widths in the column direction; the number of such entries must equal the number of rows comprising the finite-difference grid. As with the *delr* vector, elements can wrap around to the next line if desired.

Figure 2.7 demonstrates a shorthand method by which cell row and column widths can be supplied in the grid specification file. Using this method, a list of *n* identical entries of magnitude *b* can be replaced by the term `n*b`.

2.9 Integer Array File

As the name implies, an integer array file holds a structured MODFLOW/MT3D-compatible integer array. Such a file can be formatted or unformatted, the latter being a form of binary data storage which, in general, conserves disk space. The protocol for storing unformatted integer array files is such that they are capable of being read by the MODFLOW utility subroutine U2DINT which possesses an unformatted read option.

A formatted integer array can be software-generated or written manually using a text editor. Figure 2.9 shows the contents of a small integer array file.

```
10  8
3  3  3  3  3  3  3  4  4  4
3  3  3  3  3  3  4  4  4  4
2  3  3  3  3  4  4  4  4  4
2  2  2  3  4  4  4  4  4  4
2  2  2  2  2  5  5  5  5  5
2  2  2  5  5  5  5  5  5  5
2  2  2  5  5  5  5  5  5  5
2  2  2  2  2  5  5  5  5  5
```

Figure 2.9 Contents of a formatted integer array file.

The first line of a formatted integer array file contains two entries, viz. the number of columns and rows, respectively, in the model finite-difference grid. Note that these two quantities are listed in reverse order to their appearance in a grid specification file. *Note also that if `colrow` is set to "no" in the settings file `settings.fig`, then this line must be omitted.*

Next follows the integer array. The array is listed row by row starting from the top of the grid. *ncol* integers must be supplied for each row, where *ncol* is the number of columns in the finite-difference grid. The listing of row elements can wrap around onto the next line if necessary; however the elements of each new row must begin on a new line. Individual elements must be separated by spaces, tabs or a comma.

Any Groundwater Data Utility program that reads or writes an integer array first inquires of the user the name of the file in which the array resides or to which it must be written. If the filename so supplied possesses an extension of "`INF`", the utility assumes that the file is formatted (“`INF`” stands for “<u>IN</u>teger <u>F</u>ormatted”). However if the user supplies a filename with an extension of “`INU`” the relevant file is assumed to be unformatted (“`INU`” stands for “<u>IN</u>teger <u>U</u>nformatted”). If the supplied filename possesses a different extension from both of these, the user is asked specifically whether the file is formatted or unformatted so that it can read or write the file in the appropriate manner.

2.10 Integer Array Table File

An integer array table file holds an integer array written in tabular form. It holds either two or three columns of data. The first two items on each line of a three-column integer array table file list a cell row and column number respectively. The third item holds the integer array value pertaining to the cell whose row and column numbers comprise the previous entries on the line. Line entries can be tab, space or comma-delimited. Figure 2.10 shows part of an integer array table file.

```
 21      24      1
 21      25      1
 21      26      1
 21      27      1
 21      28      1
 21      29      6
 21      30      6
 21      31      6
 21      32      6
 21      33      6
 21      34      6
 21      35      6
 21      36      6
 21      37      1
 21      38      1
 21      39      1
 21      40      1
 21      41      1
 21      42      1
 21      43      1
 21      44      1
 21      45      1
```

Figure 2.10 Extract from a three-column integer array table file.

As the name implies, each line of a two-column integer array table file possesses two items. The first item on each line is a cell number whereas the second item is the integer array value pertaining to the cell whose cell number appears as the first line item. A cell number is a means of cell identification obtained by counting cells row by row from the top left of the finite difference grid.

It is not necessary that every cell contained within a finite-difference grid be referenced in an integer array table file; those Groundwater Data Utilities which read such files supply a default value for missing elements.

The principal use of an integer array table file is in exchanging integer array data with a Geographical Information System (GIS). An integer array can be uploaded from a model to a GIS as, for example, a MIF/MID file combination, a three-column integer array table being housed in the MID file. In a GIS, array elements within this table (final column only) can be altered in accordance with geographical data contained within other information layers covering the same study area. Integer array data edited in this fashion can then be downloaded as a text file such as is shown in Figure 2.10 and formatted as an integer array using pertinent Groundwater Data Utility programs.

2.11 Pilot Points File

A pilot points file is very similar to a bore coordinates file; however, in the case of a pilot points file, the integers in the fourth column refer to zones within a model domain rather than to layers. Furthermore, the file contains a fifth column listing data pertaining to each pilot point. Figure 2.11 illustrates a pilot points file.

```
pp1  5045.644  5450.288  1 1.23
pp2  6481.347  4807      1 0.89
pp3  7455.778  5622.2    1 2.34
pp4  6809.895  6832.13   1 2.19
pp5  9629.16   6862.751  1 1.98
pp6  10443.17  6133.803  1 0.23
pp7  12172.05  6396.549  1 0.32
pp8  11806.35  7939.032  1 1.03
pp9  9517.249  4838.306  2 10.2
pp10 10430.28  4825.698  2 13.6
pp11 10583.86  3666.904  2 9.43
pp12 8686.321  3047.161  2 8.32
pp13 6894.65   3704.822  2 6.43
pp14 5620.683  2736.789  2 10.4
```

Figure 2.11 A pilot points file.

The first entry on each line of a pilot points file is the "pilot point identifier"; like a bore identifier, this should be of 10 characters or less in length and is case-insensitive. Then follow the easting and northing of the pilot point. The next entry is an integer, this normally identifying the zone within a model domain in which spatial interpolation is affected by the value assigned to the pilot point; such zonation is normally defined by a MODFLOW-compatible integer array. The final entry on each line of a pilot points file is the value assigned to the pilot point. For those members of the Groundwater Data Utilities which read a pilot points file, these values are used for spatial interpolation to the cells of the finite difference grid.

Pilot points are a useful device for assigning properties to a model domain. As such, they are a useful complement to PEST's predictive analysis and/or regularisation capabilities. When used in this way, data in the final column of a pilot points file may represent hydraulic conductivity or some other aquifer property.

Entries on each line of a pilot points file should be space or tab-delimited.

2.12 Real Array File

As the name implies, a real array file holds a structured MODFLOW/MT3D-compatible real array. Such a file can be formatted or unformatted, the latter being a form of binary data storage which, especially in the case of real arrays, conserves disk space. The protocol for storing unformatted real array files is such that they are capable or being read by the MODFLOW utility subroutine U2DREL. Note that unformatted file storage is compiler-specific. Thus an unformatted file written by a program compiled with a certain FORTRAN compiler may not be readable by a program compiled with another FORTRAN compiler.

A formatted real array can be either software-generated or written manually using a text editor. Figure 2.12 shows the contents of a small formatted real array file.

```
10  8
3.567   3.567   3.567   3.567   3.567   3.567   3.567   4.539
4.539   4.539
3.567   3.567   3.567   3.567   3.567   4.539   4.539   4.539
4.539   4.539
2.34e-1 3.567   3.567   3.567   3.567   4.539   4.539   4.539
4.539   4.539
2.34e-1 2.34e-1 2.34e-1 3.567   4.539   4.539   4.539   4.539
4.539   4.539
2.34e-1 2.34e-1 2.34e-1 2.34e-1 2.34e-1 5.1     5.1     5.1
5.1     5.1
2.34e-1 2.34e-1 2.34e-1 5.1     5.1     5.1     5.1     5.1
5.1     5.1
2.34e-1 2.34e-1 2.34e-1 5.1     5.1     5.1     5.1     5.1
5.1     5.1
2.34e-1 2.34e-1 2.34e-1 2.34e-1 2.34e-1 5.1     5.1     5.1
5.1     5.1
```

Figure 2.12 Contents of a formatted real array file.

The first line of a formatted real array file contains two entries, namely. the number of columns and rows, respectively, in the model finite-difference grid. Note that these two quantities are listed in reverse order to their appearance in a grid specification file. *Note also that if `colrow` is set to "no" in the settings file `settings.fig`, then this line must be omitted.*

Next follows the real array. The array is listed row by row starting from the top of the grid. *ncol* numbers must be supplied for each row, where *ncol* is the number of columns in the finite-difference grid. While the listing of row elements can wrap onto the next line if necessary, the first element of each new row must begin on a new line. Individual elements must be separated by spaces, tabs or a comma.

Any utility that reads or writes a real array first inquires of the user the name of the file in which the array resides, or to which it must be written. If the filename so supplied possesses an extension of "`REF`", the utility assumes that the file is formatted ("`REF`" stands for "REal Formatted"). However if the user supplies a filename with an extension of "`REU`" the relevant file is assumed to be unformatted ("`REU`" stands for "REal Unformatted"). If the supplied filename possesses a different extension from both of these, the program asks the user specifically whether the file is formatted or unformatted so that it can read or write the file in the appropriate manner.

2.13 Real Array Table File

A real array table file holds a real array written in tabular form. It holds either two or three columns of data. The first two items on each line of a three-column real array table file list a cell row and column number respectively. The third item holds the real array value pertaining to the cell whose row and column numbers comprise the previous entries on the line. Line entries can be tab, space or comma-delimited. Figure 2.13 shows part of such a real array table file.

```
21      24      1.3445
21      25      1.3445
21      26      1.3445
21      27      1.3445
21      28      1.3445
21      29      6.5641
21      30      6.5641
21      31      6.5641
21      32      6.5641
21      33      6.5641
21      34      6.5641
21      35      6.5641
21      36      6.5641
21      37      11.432
21      38      11.432
21      39      11.432
21      40      11.432
21      41      11.432
21      42      11.432
21      43      11.432
21      44      11.432
21      45      11.432
```

Figure 2.13 Extract from a three-column real array table file.

As the name implies, a two-column real array table file possesses two columns of data. The first item on each line is a cell number whereas the second item is the real array value pertaining to the cell whose cell number appears as the first line item. A cell number is a means of cell identification obtained by counting cells row by row from the top left of the finite difference grid.

Note that it is not necessary that every cell contained within a finite-difference grid be referenced in a real array table file. Any Groundwater Data Utility program which reads such a file supplies a default value to uncited cells.

The principal use of a real array table file is in exchanging real array information with a Geographical Information System (GIS). A real array can be uploaded from a model as, for example, a MIF/MID file combination, a three-column real array being housed in the MID file; in a GIS the array elements within this table (final column only) can be altered in accordance with geographical data contained within other information layers covering the same study area. Real array data edited in this fashion can then be downloaded as a text file such as is shown in Figure 2.13 and reformatted as a real array again using pertinent Groundwater Data Utility programs.

2.14 Settings File

A settings file must be present within any directory from which certain of the Groundwater Data Utilities are invoked. The utilities which require it are those that read and/or write date and time information and those which read and/or write MODFLOW-compatible real and/or integer arrays. A settings file, which must possess the name `settings.fig`, is illustrated in Figure 2.14.

```
date=dd/mm/yyyy
colrow=yes
```

Figure 2.14 A settings file.

A settings file requires two lines, one informing the Groundwater Data Utilities of the protocol for expressing dates, and the other informing them of the convention to use for storage of formatted, MODFLOW-compatible, real and integer arrays. These lines can be written to the settings file in either order.

The first line appearing in Figure 2.14 informs the utilities that, in expressing dates in all pertinent input and output files, as well as in all pertinent terminal and screen input/output, the date is to precede the month. However if the first entry in Figure 2.14 were written as "`date=mm/dd/yyyy`", then the month must precede the day in representations of date.

As was explained in Sections 2.10 and 2.12, formatted files containing real and integer arrays can optionally begin with a one-line header containing the number of columns and rows (in that order) in the finite difference grid. (This header is expected by certain commercial MODFLOW graphical user interfaces.) If `colrow` is set to "`yes`" in the settings file, then this header is expected on any real or integer array read by any member of the Groundwater Data Utility suite. Alternatively, if `colrow` is set to "`no`", members of the Groundwater Data Utilities will expect formatted real and integer arrays to contain no such header, and will write formatted real and integer array files in which the header is omitted.

2.15 Structure File

A structure file stores parameters that are associated with one or more variograms assigned to one of more "geostatistical structures". Each such geostatistical structure can contain up to 5 nested variograms and a nugget. Each such structure is normally assigned to a particular zone within the model domain, different zones normally being differentiated from each other on the basis of different values in a model-compatible integer array.

Figure 2.15 depicts a structure file.

```
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
  NUMVARIOGRAM  1
  MAXPOWERVAR 10000
  VARIOGRAM var3 .005
END STRUCTURE


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

Figure 2.15 A structure file.

A structure file is made up of any number of separate incidences of two different types of entity. These entities are the “structure” entity and the “variogram” entity. The latter entity specifies a variogram. Any number of such variograms can be nested to form a geostatistical structure. Kriging within each zone of the model domain, as carried out by programs such as PPK2FAC and FAC2REAL, is based on the geostatistical properties of an area as defined in a structure. Any number of structures and variograms can be cited in a structure file.

Complete specifications of a structure file are presented in the documentation to programs PPK2FAC and PPK2FAC3D in Part B of this manual. See also Section 5 of the present document where the use of pilot points and geostatistical methods in model parameterisation and calibration is fully discussed.

2.16 MODFLOW-USG Specification Files - General

MODFLOW-USG was released in May 2013. Its introduction required the definition of a new grid specification file which provides geographical details of grid construction. This can then be used by utilities which act as pre- and post-processors for MODFLOW-USG.

A particularly powerful pre-processing program for MODFLOW-USG is PLPROC. This can be used in place of many of the programs comprising the Groundwater Data Utilities when using pilot points as a parameterization device for a MODFLOW-USG model.

Some members of the Groundwater Data Utilities suite can perform processing functions that enhance and facilitate the use of MODFLOW-USG, either on its own or in conjunction with PEST. Some of these utilities need to read a MODFLOW-USG grid specification file. These include programs such as USG2VTK which facilitates display of MODFLOW-USG grid details, model properties, and calculated system states.

The grid specification file for an unstructured grid is far more complex than for a structured grid. The lack of grid structure precludes omission of the coordinates of the vertices of each model cell. Hence the locations of these vertices must be specified in the grid specification file, together with the node that is associated with a model cell.

An additional complication with MODFLOW-USG is the fact that not only can this model employ a grid for groundwater flow simulation; it can also simulate flow in a connected linear network (referred to as a “CLN” in MODFLOW-USG parlance). Graphical representation of the components of a CLN also requires specification of network vertices, and of associations between nodes and vertices.

The following two sections provide specifications for a MODFLOW-USG grid specification file and for a MODFLOW-USG CLN specification file.

2.17 MODFLOW-USG Grid Specification File

Specifications for a MODFLOW-USG grid specification file are now provided. Those for a MODFLOW-USG CLN specification file are presented in the following section.

Line 1:

"UNSTRUCTURED GWF"

Note: If "GWF" is omitted, then it is assumed.

Line 2:

nnode nlay iz ic

where:

nnode	*is the number of nodes in the grid*
nlay	*is the number of layers in the model*
iz	*is 1 if elevations of node and mesh element vertices are supplied; 0 otherwise*
ic	*is 1 if the cell specifications associated with each node are supplied; 0 otherwise*

Note. All utilities documented herein, as well as PLPROC, require that both iz and ic be 1. Options associated with values other than 1 are not presented below. If these are omitted from the grid specification file they are assumed to be 1.

A list of vertex coordinates is now provided. These vertices will be cited by index later in this file where mesh geometric details are provided. The indices of vertices are defined implicitly by their positions in the following list. Numbering is assumed to begin at 1.

Line 3:

nvertex	*is the number of element vertex definitions to follow*

NVERTEX next lines:

x, y, z

where:

x, y and z	*are the coordinates of each vertex*

NNODE next lines:

inode x y z lay m (ivertex(i),i=1,m)

where:

inode	*is a node number (these must be supplied in increasing order starting from 1)*
x, y and z	*are node coordinates*
lay	*is the layer number of the node*
m	*is the number of vertices defining the three-dimensional element associated with the node*
ivertex	*are vertex indices defined with reference to the vertex list provided above*

Figure 2.16 Specifications of a MODFLOW-USG grid specification file.

2.18 MODFLOW-USG CLN Specification File

Specifications for a MODFLOW-USG CLN specification file are now provided. Those for a MODFLOW-USG grid specification file are presented in the preceding section.

Line 1:
"STRUCTURED CLN"

Line 2:
nnode ncln
where:

nnode	*is the number of CLN nodes*
ncln	*is the number of CLN segments*

A list of vertex coordinates is now provided. These vertices will be cited by index later in this file where CLN geometric details are provided. The indices of vertices are defined implicitly by their positions in the following list. Numbering is assumed to begin at 1.

Line 3:

nvertex	*is the number of element vertex definitions to follow*

NVERTEX next lines:
x, y, z
where:

x, y and z	*are the coordinates of each vertex*

NNODE next lines:
inode x y z numseg (iseg, i=1,numseg), ((nvert(i), vertnum(j),j=1,nvert), i=1,numseg)
where:

inode	*is a node number*
x, y and z	*are node coordinates*
numseg	*is the number of CLN segments on which the node lies*
iseg	*is a segment index (can be 1 to NCLN)*
nvert(i)	*is the number of vertices that comprise the polyline (must be at least 2) that the node represents*
vertnum(i)	*is the vertex number – nvert(i) of these must be supplied for a polyline*

Figure 2.17 Specifications of a MODFLOW-USG CLN specification file.

The above specifications assume the following.

- A node represents a linear or polylinear feature. This feature may be represented by a single chord (which lies between two vertices), or by a polyline representing *N* chords (lying between N+1) vertices. It is these vertices which can be plotted to represent the node. (Optionally their colour will represent a node's property when it is displayed.)
- A node can lie on more than one CLN segment – and hence represents the intersection of CLN segments.

The first part of the CLN specification file provides vertex coordinates. The next part provides vertex-node associations. Vertices define polylines (or a single line). More than one polyline (line) is associated with a node only if that node lies on more than one CLN segment, and hence lies on the intersection of those segments.

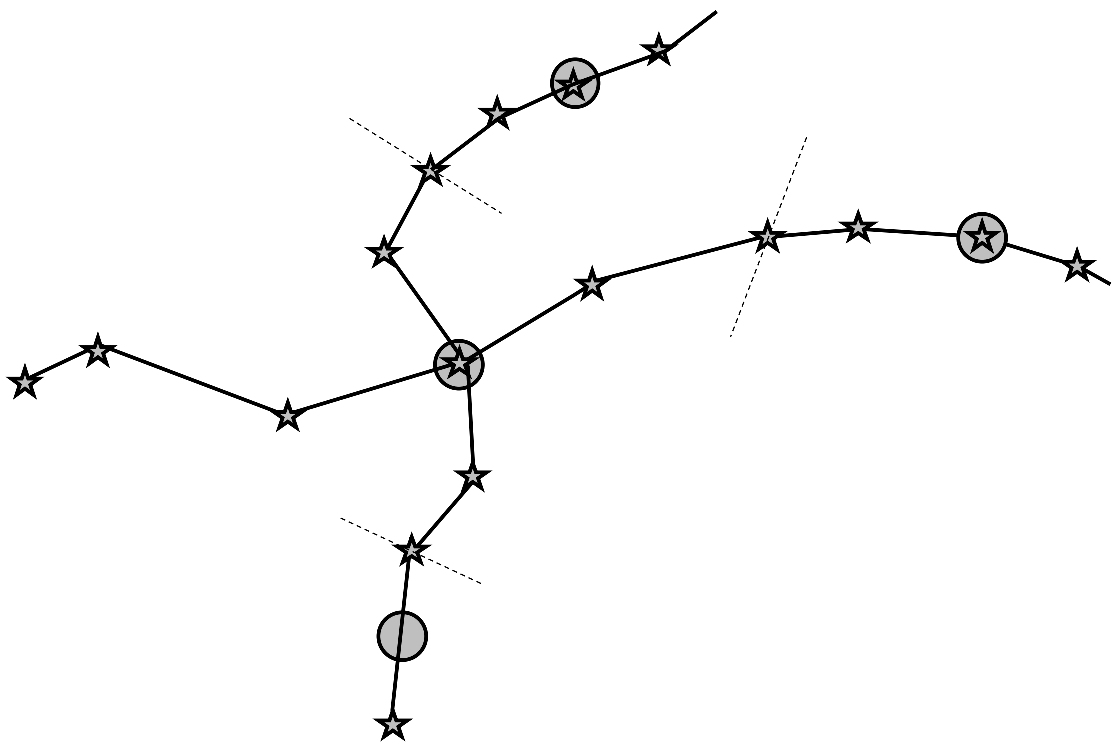

Figure 2.18 A node lying at the intersection of two segments.

Nodes in Figure 2.18 are represented as circles; vertices are represented by stars. One particular node in the above figure lies at the intersection of two CLN segments. Hence NUMSEG for that node is 2. It is associated with two intersecting polylines. For the near-vertical polyline with which it is associated NVERT is 5 as the four chords which form it are defined by 5 vertices. NVERT for the near horizontal polyline is 6.

Hence it is assumed that even though each node in a CLN network represents a “linear” feature, in fact the feature may be curved for display purposes. In most cases however a node will, in fact, represent a strictly linear feature. The latter will be represented by two vertices. The node will lie at the centre of the feature. Hence the feature will not “bend”.

2.19 MODFLOW-USG Nodes-in-Layer File

Some of the utility programs documented in Part C of this manual need to know how many nodes lie within each layer of a MODFLOW-USG grid. While this information can be obtained from a grid specification file, such a file may not be available. A far simpler file to prepare is a nodes-in-layer file. As the name suggests, this file lists the number of nodes within each layer of a MODFLOW-USG grid.

An example of a nodes-in-layer file is provided in Figure 2.19.

```
5
12472  12472  784 784  784
```

Figure 2.19 An example of a nodes-in-layer file.

The first line of a nodes in layer file must contain a single entry, this being the number of layers in the unstructured grid. Then, starting on the following line, must follow a one dimensional array which lists the number of nodes in each layer. This can be copied from the NODELAY array provided in the unstructured grid discretization file (i.e. DISU file). This array is read in free field format. Hence it can wrap into the next line if required. Multiple elements can be represented with the "n*" protocol if desired. Hence the second of the above lines could be written as "2*12472 3*784".

2.20 Node-to-Bore Interpolation File

A number of Groundwater Data Utility programs interpolate MODFLOW-USG-calculated quantities such as heads/drawdowns to arbitrary points in space. The former are associated with the nodes of a MODFLOW-USG unstructured grid. Often the latter are the sites of observation wells. Some of these utility programs require that the factors through which interpolation is implemented from MODFLOW-USG nodes to these points in space be supplied by the user. These factors must be supplied in a node-to-bore interpolation file. An example of such a file is provided in Figure 2.20.

```
well_324  4  34543  0.211    34544  0.433  74223   0.123  74223  0.233
well_325  3  67849  0.342    67849  0.233  84432   0.425
etc
```

Figure 2.20 Part of a node-to-bore interpolation file.

The first entry on each line of a node-to-bore interpolation file must be the name of a well (or the name of a point). This name must be 20 characters or less in length. A positive integer must follow this name, this indicating the number of nodes used for interpolation to that point. Suppose that this number is *N*; then *N* pairs of numbers must follow. The first item of each pair must be a MODFLOW-USG node number (a positive integer); the interpolation factor associated with that node (a real number) must follow that.

Note the following.

- Entries in a node-to-bore interpolation file should be space delimited.
- Interpolation factors must sum to 1.0.
- As few as one node can be used to interpolate to any one point in space; there is no upper node number limit.
- Nodes employed for interpolation to any one point do not need to belong to the same MODFLOW-USG grid layer.

- Either groundwater flow (GWF) or connected linear network (CLN) nodes can be used for interpolation to any particular point, but not both of these together.

- Where CLN nodes are cited, the numbers used to specify these nodes must be the same as those employed for this purpose in the CLN package input file, i.e. the IFNO variable pertaining to each node.

2.21 Node Data Table File

A node data table file contains two or more columns of numbers. The first column must list, in sequential order, all of the nodes of an unstructured grid. These must be numbered from 1 to *N* where *N* is the total number of nodes in the grid. Note that the grid can be a complete groundwater flow (GWF) grid, or a connected linear node (CLN) grid. In either case the numbers must be sequential and start at 1.

Columns after the first contain data pertaining to nodes of the grid. Data within each column can be integer or real.

Each column must be preceded by a text header of 15 characters or less. The header for the first column must be "NODE_NUMBER". Headers for subsequent columns are arbitrary, but must include no spaces.

Both column headers and data must be space-delimited.

An example of the first part of a node data table file is provided in Figure 2.21.

```
NODE_NUMBER         elevation     conductance
1                    12.3456      1.14563e3
2                    14.2343      2.31133e4
3                    18.3353      2.34234e4
4                    17.4345      8.49454e4
5                    19.4345      8.32343e3
etc
```

Figure 2.21 First part of a node data table file.

3. Some Common Tasks

3.1 Introduction

The purpose of this section is to present an outline of the procedures to follow in order to accomplish a few of the tasks which commonly face those engaged in groundwater data processing and modelling. The tasks are arranged in no particular order; nor is the list of tasks complete. It is intended that only a brief overview be presented of some of the possibilities afforded through use of the Groundwater Data Utilities. For details of the use of any particular program, see Parts B and C of this manual.

3.2 Preparing Water Level Contours

To generate a contour map of water levels on or about a certain date, program SMP2INFO should be used to interpolate the water level data contained within a bore sample file to the date for which the contour map is desired. SMP2INFO generates a bore information file in which bore coordinates are included. This file can be presented to a contouring program such as SURFER as an *xyz* data file for gridding and subsequent contouring.

The bore information file generated by SMP2INFO can also be used to post and label the bore locations on the contour map. Labels can consist of bore identifiers, or of borehole water levels interpolated to the user-specified date.

To enhance the quality of the contour map, the grid file produced by SURFER at the data interpolation stage should be blanked at the aquifer boundaries (if they are clearly defined) or along a user-specified demarcation line where it is judged that contour lines are too distant from the nearest bores to be believable. There are a number of ways in which an appropriate SURFER blanking file can be constructed for this purpose. One way is to use the SURFER “Digitize” facility to construct a blanking file by digitizing directly from the screen. Another way is to construct a blanking file based on the boundary of the active part of a MODFLOW model grid. Firstly an integer array should be constructed in which all active and fixed head cells are assigned the same non-zero value, and all inactive cells are assigned a value of zero (this can easily be accomplished through slight modification of a model activity array using the array editing capabilities of many MODFLOW graphical user interfaces). Then program ZONE2BLN can be used to construct a blanking file that will effectively blank any part of the SURFER grid that lies within the inactive part of the finite difference grid, or outside of the finite-difference grid altogether. As the active part of a finite-difference grid normally coincides with the aquifer, a blanking file produced in this manner blanks all contours where the latter are drawn outside of the aquifer.

It should be noted that a bore sample file can contain any type of borehole data, not just water level data. Hence contour maps of other time-variant aquifer characteristics such as chemical concentration can be produced in an identical manner to that described above for the production of water level contours.

3.3 Preparing Bore Hydrographs

In addition to providing a basis for the construction of aquifer iso-characteristic maps in the manner described above, water level or water quality data housed within a bore sample file can be extracted and plotted against time. Program SMP2HYD provides the functionality necessary to extract data for user-specified bores from a bore sample file, and to format it in a manner suitable for use by plotting packages such as GRAPHER, EXCEL, etc.

SMP2HYD can extract data for one or many bores. It can extract all of the historical data pertaining to a given bore, or can simply extract data between two specified dates and/or times. SMP2HYD’s output files list borehole sample data together with time elapsed from a reference date and time; thus the data is in a form ready for immediate plotting. The units of elapsed time are user-selectable, as are the time and date of the temporal reference.

Using the functionality available through most commercial plotting packages, borehole data extracted by SMP2HYD can be plotted to any scale; multiple hydrographs can be placed on the same graph for easy comparison between bores if desired. For most packages the time axis can be annotated with either elapsed time or with actual dates and/or times.

3.4 Extracting and Manipulating Borehole Pumping Data

The amount of water pumped from user-specified bores between two-user specified dates can be calculated using program PMP2INFO. PMP2INFO obtains its data from a bore pumping file of the type illustrated in Figure 2.4. The dates and times which define the beginning and end of a particular time interval need not coincide with borehole measurement times, for PMP2INFO interpolates between the latter times to the user-specified time interval endpoints. Pumped water units in the PMP2INFO output file are the same as those used by the bore pumping file.

The bore information file written by PMP2INFO tabulating borehole extracted volumes includes the geographical coordinates of each extraction (or injection) bore. Hence the PMP2INFO output file can be used by mapping software for posting pumping information on a map. For example a symbol can be placed on the map at each bore location, the size of the symbol being a function of the amount of water pumped.

PMP2INFO output files can also be used for transferring pumping data to a model, as discussed in the next section.

3.5 Importing Borehole Pumping Data into a Model

The assimilation of borehole pumping data into a model is one of the most time-consuming tasks in model preparation, especially for a transient model where such data must be supplied for each stress period. In irrigation areas where pumped bores may number in the hundreds, software support for this task is essential.

When using the Groundwater Data Utilities, the incorporation of borehole pumping data into a model is a two-stage process. First a bore information file must be produced for each stress period using PMP2INFO in the manner described in the previous section. For each stress period the appropriate bore information file will list the amount of water pumped during that period expressed in units used by the bore pumping file from which the data was extracted. Note that before this first processing stage can be implemented, a user must be aware of the date and time of the beginning and end of each model stress period.

The second step involves the assignment of borehole extraction rates to individual model cells. For MODFLOW modelling, borehole extraction can be supplied either through the well package or, if all extraction is from the same layer, as equivalent negative recharge through the recharge package. In either case, data can be transferred from a bore information file to the model using program PT2ARRAY.

PT2ARRAY reads a bore information file and a bore coordinates file. On the basis of the information contained in the former file and the coordinates contained in the latter file, it assigns values to those elements of a MODFLOW-compatible real array for which respective grid cells contain one or more bores. The values assigned to these elements can be modified by a scaling factor to take account of units conversion. A MODFLOW graphical user interface can then read the real array either as a recharge array, or, for some preprocessors, as a “well array”.

3.6 A Rudimentary Model-GIS Interface

There are a number programs within the Groundwater Data Utilities which provide a mechanism for integer and real array data exchange between MODFLOW/MT3D and a GIS. Real and integer arrays play a pivotal role in the parameterization and calibration of MODFLOW, MT3D and other finite-difference models. Their importance in data exchange between models, GIS, model preprocessors and visualisation software is further enhanced by the fact that most MODFLOW graphical user interfaces provide the ability to import and export MODFLOW/MT3D data arrays.

Programs INT2MIF and REAL2MIF each produce files in MAPINFO Interchange Format, the former based on a MODFLOW/MT3D-compatible integer array, the latter based on a MODFLOW/MT3D-compatible real array. The “MIF” file supplies MAPINFO (or any other GIS which reads this type of file) with the geographical information which it needs in order to plot the finite difference grid (or a subset of the finite difference grid defined by a “window” integer array) on a map; the respective “MID” file contains a table listing the value of each element of the integer or real array, each element being identified by its row and column number. This table can be edited within the GIS, either on a cell-by-cell basis, or using GIS functionality based on the properties of other geographical layers covering the same area. The edited table can then be downloaded, converted to integer or real array format using program TAB2INT or TAB2REAL, and imported into a model graphical user interface, or used by MODFLOW or MT3D directly.

3.7 Making Pictures of the Model Grid

Programs GRID2DXF and ZONE2DXF produce DXF files of a subset of the finite difference grid, and of the outlines of multi-cell zones within the finite difference grid respectively. The latter are defined in terms of groups of cells of constant value contained within a MODFLOW/MT3D-compatible integer array. The resultant DXF files can be imported into most mapping software where lines can be coloured and thickened as desired. A pleasing effect can be obtained by drawing the active part of the finite-difference grid using fine lines and superimposing internal grid zonation boundaries using heavier lines. Another useful context for the plotting of grid zonation boundaries is in the reporting of grid zonation values; the latter can be written as text within the grid zonation schematic.

Functionality also exists within the Groundwater Data Utilities for generating SURFER “blanking” files of the finite difference grid, and of the zonation contained within this grid in a manner analogous to the production of DXF files of these same features. Programs GRID2BLN and ZONE2BLN are identical to GRID2DXF and ZONE2DXF except that they generate blanking files instead of a DXF files.

3.8 Interpolating Model-Generated Heads to Bore Sites

The data contained within MODFLOW and MT3D-generated unformatted head, drawdown, concentration and other two-dimensional arrays can be interpolated to the sites of bores using program MOD2SMP. MOD2SMP constructs a bore sample file from a set of MODFLOW/MT3D-generated head, drawdown, concentration or other two-dimensional arrays (these are written in unformatted form during each model run). During construction of the bore sample file, the model results arrays are interpolated to a set of user-specified points, these points usually being the sites of bores from which field measurements were taken. Sample dates and times written to the bore sample file correspond to model output times; the user is asked for a reference date and time corresponding to the beginning of the model run.

Once a bore sample file has been constructed on the basis of model-generated data interpolated to bore sites, direct comparison of field and model data is a simple task. In particular, program SMP2INFO can be used to construct contour maps of field and model data for any user-specified date. Program SMP2HYD can be used in the construction of bore hydrographs for both model and field data. Using software such as GRAPHER, model and field hydrographs for one or a number of bores can be superimposed for ease of comparison.

3.9 Constructing a Section through a Model

Using program SECTION a transect of arbitrary complexity can be constructed through the model domain. SECTION produces files that can be plotted by software such as GRAPHER and EXCEL; its role is to extract and interpolate model data upon which the plot is to be based.

SECTION uses a similar spatial interpolation algorithm to that used by MOD2SMP; it interpolates data contained in MODFLOW/MT3D-compatible real arrays to a regular set of points of user-defined density along a transect line with user-defined endpoints. It can assimilate data from one or many arrays. Thus if it is supplied with a land surface elevation array, aquifer top and bottom elevation arrays and a model-generated heads array, the file written by SECTION can be used to generate a picture of the aquifer showing its top and bottom, as well as the land surface and the piezometric surface, the latter generated by the model. By incorporating the top and bottom elevations of other layers and/or model-generated heads at other model output times, a complex, yet instructive, picture of subsurface conditions can be constructed.

3.10 Using SURFER to Construct an Initial Conditions Array

A number of MODFLOW/MT3D pre- and postprocessors include the functionality whereby aquifer data obtained on a point-by-point basis can be spatially interpolated to the finite difference grid in order to produce initial condition arrays. However interpolation options are often limited so that the array of initial conditions is often far from satisfactory. This is of particular concern in the case of initial heads arrays; a model may spend the first few time steps of its simulation in simply moving water between cells as rapidly as possible in order to compensate for an initial heads array that has unnecessary "bumps", or that causes water to temporarily flow in unusual directions simply as an artefact of the interpolation scheme used in its construction.

If a model finite difference grid is uniform, SURFER can be used to interpolate point data to the finite-difference grid cell centres. SURFER possesses many more interpolation options than do most model preprocessing packages, including the "minimum curvature" method which (in the author's experience) often results in an array that functions well as an initial conditions array. If a SURFER grid is constructed such that *x* and *y* direction grid lines intersect at finite-difference grid cell centres, then the contents of a SURFER grid file can be used directly as a model real array. However the grid file (which must be stored in ASCII form) must first be translated into model-compatible real array format before importation into a MODFLOW preprocessor; this can be accomplished using program SRF2REAL.

A problem occurs where the grid is uniform but is not oriented with its row direction east-west and its column direction north-south, for a SURFER grid can only be thus oriented. In this case the model grid must be rotated so that it can be simulated by a SURFER grid. However not only must the model grid be rotated, but the data used to construct the initial conditions array must also be rotated. Program ROTDAT carries out the latter function, transforming point coordinates data contained in an *xyz* file in a manner equivalent to rotating all data points about the top left corner of the finite difference grid, and assigning the latter point *x* and *y* coordinates of (0,0). If the rotated and translated data is then read by SURFER and interpolated onto a grid which is equivalent to the finite difference grid in terms of its number of rows and columns, but for which the top left intersection point (ie. node) is situated at (*delr/2.0,-delc/2.0*), the grid file thus produced can be translated into a real array file by SRF2REAL for immediate model use. Here *delr* is the (uniform) model cell width in the row direction while *delc* is the (uniform) model cell width in the column direction.

Note that if a grid is nonuniform, SURFER can still be used in the production of initial conditions arrays, though now a two-step process is required. The first step is to grid the point data using SURFER to a fine grid which is large enough to cover the entire model area (no grid rotation or translation is required). The SURFER grid can then be stored as an *xyz* file to be subsequently used by a model preprocessor's interpolator for the production of a model initial conditions array. The fact that the second stage interpolator may not be very good is of little consequence because *xyz* data points are close to all model cell centres, thanks to first stage interpolation to a dense grid carried out by SURFER.

3.11 Using SURFER to Display Model Results

Section 3.8 discusses the means whereby model results can be interpolated to bore sites for subsequent display as either individual hydrographs or as contours. However if a model grid is uniform, a model-generated real array can be translated into a SURFER grid file using program REAL2SRF. This grid file can be directly read by SURFER as if it had constructed it itself; thus no intermediate interpolation is required and SURFER is able to construct contours on the basis of "raw" model data.

If the uniform model grid is oriented with its row direction east-west and its column direction north-south, map data can be directly overlain on the contour map generated by SURFER on the basis of the grid file built by REAL2SRF. However if the grid row direction is oriented at an angle other than east-west, REAL2SRF writes the SURFER grid file using coordinates that effectively rotate and translate the grid until the row direction is oriented east-west and the top left corner of the grid is at the point (0,0); this rotation and translation operation is necessary because a SURFER grid can only be oriented with its *x* direction pointing east and its *y* direction point north. Any map data that is to be overlain on a contour map built from this grid file must likewise be rotated and translated. The functionality to undertake this data rotation and translation is available within the Groundwater Data Utilities. The contents of a DXF file can be appropriately transformed using program ROTDXF. Program ROTBLN transforms the contents of a SURFER blanking file. *xyz* data can be transformed prior to posting on the SURFER-generated contour map using program ROTDAT.

Once SURFER has produced its contour map, and map data (transformed as above) has been overlain on it, the composite picture should then be rotated to its true orientation using the SURFER "Arrange/Rotate" menu item. Note, however, that axis labels will still reflect the artificial, rotated and translated coordinate system, and should thus be eliminated from the plot.

If a model finite difference grid is not uniform the above method cannot be used to export model results to SURFER. In this case, if it is desired that SURFER be used to display model results, SURFER should be presented with an *xyz* file containing the model-calculated head (or concentration, etc.) at each active cell centre. SURFER can then grid this data using one of its internal gridding algorithms prior to displaying the results. The interpolation method will not matter too much as no SURFER grid point will be too far from a model-specific *xyz* point (except where the SURFER grid over-reaches the active part of the model grid; in this case over-reaching segments should be blanked using one of the methods described in Section 3.2 above).

The construction of an *xyz* file of model results at all active cell centres is a two-step process. First a file tabulating the coordinates of active cell centres must be constructed; next model results must be interpolated to these points. The first step can be accomplished using program GRID2PT; the second can be accomplished using program MOD2SMP followed by SMP2INFO.

3.12 Extracting Model-Generated Arrays

MODFLOW and MT3D generate unformatted data files containing two-dimensional arrays of head, drawdown, concentration, etc data. These files cannot be inspected with a text editor as data are represented in binary format.

Programs MANY2ONE and GETMULARR have been designed to extract individual arrays from MODFLOW and MT3D unformatted output files and re-write them in formatted form where they can be viewed with a text editor and/or transferred to model pre- and postprocessing packages. Program ARRDET lists the contents of a MODFLOW or MT3D unformatted file. Programs that perform similar functions for MODFLOW-USG and MODFLOW 6 are document in Part C of the manual for the Groundwater Data Utilities.

3.13 Array Data Manipulation

Mention has already been made within this report of the importance of integer and real arrays in MODFLOW/MT3D pre- and postprocessing. The ability to transport such arrays between MODFLOW Graphical User Interfaces and the Groundwater Data Utilities, visualisation and mapping software such as SURFER and GIS, and the model itself is fundamental to flexible and powerful model data processing and results analysis. Because of the importance of model data arrays in the overall modelling effort, a number of Groundwater Data Utilities carry out various types of array processing and manipulation.

Program INT2REAL builds or modifies a real array based on the contents of an integer array. This is especially useful where an integer array defines spatial parameter zonation. For example an integer array may define land use units throughout a modelled catchment. The effects of different land use strategies on groundwater levels may be tested by building different recharge arrays on the basis of the integer array, recharge rates being assigned on the basis of the land usage pertaining to each zone. Using INT2REAL real arrays can be built from scratch, or an existing real array can be modified within user-specified zones.

Program REAL2INT builds an integer array from a real array either on the basis of a one-to-one correspondence between real and integer array elements, or by assigning integer array values on the basis of value ranges within the real array. The integer array can then be used in further processing or for display. For example, through the use of ZONE2DXF a picture of array zonation boundaries can be superimposed on a background map plotted using most mapping software.

Program TWOARRAY carries out mathematical operations (including partial array replacement) between two MODFLOW/MT3D compatible real arrays. All operations are such that user-specified portions of an array can remain unaffected by the manipulations.

The roles of MANY2ONE and GETMULARR in selecting individual results arrays from a MODFLOW or MT3D output file have already been discussed in Section 3.12 above.

3.14 PEST and the Groundwater Data Utilities

The present section discusses a number of ways in which programs of the Groundwater Data Utilities can be used in conjunction with PEST. However you should also read Section 5 of this document (as well as the PLPROC manual) where the use of pilot-point-based parameterisation is discussed. As is described in these documents, it is possible, using these programs, to instigate a model calibration process based on the use of pilot points, complemented by the imposition of geostatistically-based constraints on parameter values using PEST's regularisation functionality.

The Groundwater Data Utilities (or PLPROC) can be used in conjunction with PEST and its utilities to construct a powerful modelling system. The fact that these programs are executed from the command line allows them to be run from within a batch file as part of a “composite model”. This composite model can be run by PEST in order to calibrate it against field data of various types.

In most cases such a composite model will include MODFLOW and/or MT3D. Some of the Groundwater Data Utilities (or PLPROC) can be run from within the batch file as preprocessors to one or both of these models; others of these programs (or program OLPROC) can be used as postprocessors. Others can be run outside of the batch file for automatic PEST input file generation.

A MODFLOW-compatible real array can be constructed on the basis of a MODFLOW-compatible integer array using program INT2REAL, the integer array being used to define zonation for a particular model layer. The real array can then be “pasted” into an existing MODFLOW input file using programs such as REPARRAY; alternatively, it can be read by MODFLOW or MT3D itself using the EXTERNAL or OPEN/CLOSE array-reading functionality provided by these programs. Integer-real correspondences can be stored within an "integer-real correspondence file". A PEST template file is easily built from such an integer-real correspondence file. This provides an efficient mechanism for estimating parameters pertaining to certain model domain zones, for it is far easier to construct a template file based on an integer-real correspondence file than to build a template file based on a MODFLOW input file (especially where elements occurring within large arrays are in need of estimation). The first part of the composite model will then be comprised of INT2REAL (possibly followed by REPARRAY) followed by MODFLOW. In all cases the answers to command-line prompts generated by these programs should be written to a file using a text editor, and supplied to the program using command line redirection. A program such as INT2REAL is thus run from within the batch file using a command such as:-

```
int2real < int2real.in
```

where file `int2real.in` contains pre-recorded answers to INT2REAL's prompts.

Complex real array manipulation can be undertaken between generation of one or more real arrays on the basis of one or more integer arrays, and the inclusion of one or more arrays in an existing MODFLOW dataset. Such array manipulation can be carried out with program TWOARRAY which must thus be executed from within the composite model encapsulated in the model batch file.

Once MODFLOW or MT3D has been run, model-generated water levels or solute concentrations will often need to be spatially interpolated to the sites of boreholes. This can be accomplished using program MOD2SMP, the outcome of MOD2SMP's execution being a bore sample file. This operation allows field data to be compared with model data. For a more exact comparison, model-generated heads and solute concentrations can also be *temporally* interpolated to the *times* at which measurements were taken, in addition to being *spatially* interpolated to the *locations* of field measurements. This dual time-space interpolation can be carried out using program MOD2OBS; the outcome is, once again, a bore sample file. If this is done, MOD2OBS must be called from within the composite model batch file after MODFLOW and/or MT3D have been run from within this same file.

Inflows and outflows of water to/from the model domain within each of a number of user-specified zones can be tracked from time step to time step using program BUD2SMP. The outcome of this operation is, once again, a bore sample file. Using BUD2SMP inflows and outflows are recorded at times for which such information is generated by MODFLOW. If program SMP2SMP is run after BUD2SMP, another bore sample file is generated in which model inflow/outflows are *temporally* interpolated to the *times* at which field measurements are recorded in another bore sample file. Thus a direct comparison can be made between field measurements and model outputs.

If either MOD2OBS or SMP2SMP is used to build a model-generated equivalent to an existing measurement-based bore sample file, model results and field measurements can be directly compared on a one-to-one basis. Through the use of program PESTPREP, a suite of PEST input files can then be generated through which a PEST run can be undertaken in order to adjust specified model parameters until the discrepancies between model and field data are reduced to a minimum in the weighted least squares sense. Through the use of PESTPREP preparation time for a PEST run can be reduced to minutes, even where observation datasets are very lengthy. The model that is actually run by PEST is thus a batch file comprised of MODFLOW and/or MT3D together with MOD2OBS, or BUD2SMP followed by SMP2SMP (or even all three of these postprocessors), possibly in conjunction with one or a number of preprocessors as described above.

See Part C of this manual for similar programs which can be used in conjunction with MODFLOW-USG and MODFLOW 6. See also documentation of programs PESTPREP1 and PESTPREP2 in part B of this manual, as well as documentation for OLPROC.

The ability to build a composite model according to the demands of a particular modelling application, and to obtain model outputs which are both spatially and temporally interpolated to the places and times of field measurements, complemented by the ability to automate generation of PEST input files for systems of arbitrary complexity, constitutes a powerful modelling system.

3.15 Model Inflows and Outflows

Although programs such as ZONEBUDGET allow a user to obtain model-calculated water exchange rates between various user-specified zones within a model domain, the output files generated by this and other, similar, programs are not normally in a form whereby the variation of this inflow or outflow over time can be easily plotted. Such functionality is available, however, through program BUD2HYD. If cell-by-cell flow terms are recorded during a transient MODFLOW run, BUD2HYD is able to extract flow rate data generated by user-specified packages throughout the simulation time, and to format this data in a manner readily acceptable by most commercial plotting packages. This makes it easy to generate graphs of, for example, river baseflow or mine inflow against time through a model simulation period.

4. Model Parameterisation based on Pilot Points

Note: since the time of writing this section I have written a number of new programs which provide assistance in pilot point and stochastic parameterisation of a groundwater model. These include FIELDGEN, ARRAYOBS, REAL2TAB, VERTREG, GENREG, PARAM3D and others. See Parts B and C of this manual for a description of the individual programs. I'll add some details to this section when I have the time.

See also PLPROC – a "parameter list processor" which provides advanced spatial interpolation functionality for use with models such as MODFLOW-USG and MODFLOW 6 which employ an unstructured grid. It can also be used with a traditional structured MODFLOW grid.

4.1 Introduction

A number of programs supplied with the Groundwater Data Utilities can be used to assist in spatial property definition across a model domain. In doing this they can replace, or at least complement, spatial parameter definition based on zonation. When employing zones to define the distribution of some hydraulic property over a model domain, that property is assumed to be uniform within each such zone. Although zone boundaries are often defined in such a way as to coincide with mapped geological boundaries, zonation as a means of parameter definition is often unsatisfactory for at least the following reasons:

- the locations of geological boundaries are often subject to a large degree of uncertainty, especially in areas where outcrop is scarce;
- the disposition of geological units with depth is often poorly known;
- rock types that are grouped into similar geological units for the purposes of producing a map may have very different hydraulic properties, and thus may not constitute a proper grouping from a hydrogeological point of view;
- even where geological grouping is based on perceived hydraulic similarity, hydraulic property homogeneity can rarely be guaranteed.

The last of the above points is of particular importance. It is rare that a zonation based on geological characterisation does not need some refinement as the model calibration process (which can be considered as a form of data interpretation) exposes the fact that heterogeneity, in addition to that encompassed in the current zonation pattern, must prevail within the model area if model outputs are to match field observations to an acceptable degree. The problem then arises as to how much additional zonation should be introduced to the model domain, and where the boundaries of new zones should be drawn. Furthermore, once enough zones have been introduced to the model domain to allow a good fit between model outputs and field data to be achieved, the resulting zonation pattern can look artificial. Justification for the introduction of extra zones is based on the fact that the calibration process has proved that heterogeneity exists, and to thus ignore this heterogeneity would be worse than adopting a zonation pattern which is somewhat arbitrary and which, due to the fundamental nonuniqueness of the calibration problem, is probably one scheme amongst many others that would also have provided an acceptable fit between model outputs and field data.

The present section describes an alternative method of model domain parameterisation based on geostatistics and pilot points. It does not, by any means, remove the fundamental nonuniqueness of the parameter estimation problem. However the main advantage of the method is that it allows parameter definition to take place in a more flexible manner than through the use of zones. This, in turn, allows modern methods of parameter estimation and uncertainty analysis to be applied to the problem of model calibration.

When used in regularisation mode in conjunction with a parameterisation scheme that allows smaller scale hydraulic property variations to be represented than can be specified on the basis of zones alone, PEST can be asked to make a model domain (or individual zones within a model domain) only as heterogeneous as it needs to be in order to achieve an acceptable level of fit between model outputs and corresponding field data. “Regularisation constraints” necessary to achieve this “minimum heterogeneity” condition can be designed in such a way as to reflect any available geostatistical or intuitive evidence within a study area pertaining to any heterogeneity that might exist. Such evidence normally takes the form of a specification that hydraulic property differences between two points are likely to be smaller if the two points are closer together than if they are further apart. Furthermore, the capacity of the parameterisation scheme to allow small scale heterogeneity to be represented allows PEST to place such heterogeneity only at those locations where it will “do the most good” from the point of view of matching model outputs to field data.

4.2 Geostatistical Concepts

The purpose of this section is to provide the geostatistical underpinnings of the pilot-point-based parameterisation methodology implemented through a number of the Groundwater Data Utility programs documented in Parts B and C of this manual. Only limited information is provided on geostatistical methods in general. Many excellent texts on that subject are available. See, for example, Deutsch and Journel (1998); this contains a description of the GSLIB library upon which the software described herein is based.

Where pilot points are used as the basis for hydraulic property definition throughout a model domain, values of that property are first defined at the locations of the points. Hydraulic property values are then assigned to all cells within the model domain through spatial interpolation of the values assigned to the pilot points. As presently implemented, such spatial interpolation takes place using the method of kriging. As is described in any geostatistical text, this method is intended to minimise the uncertainty of the property estimate at each interpolated point, based on an assessment of the spatial variability of that property as specified by a variogram. Use of a variogram to characterise spatial heterogeneity is based on the premise that the closer are two points situated with respect to each other, the more likely are their properties to be similar, and that the degree to which this premise holds true does not vary across a study area or geological unit.

Four types of variogram are supported by programs of the Groundwater Data Utilities that implement the pilot-point-based parameterisation scheme described herein. They are as follows:-

Spherical

$$\gamma(h) = c \cdot \left[1.5\frac{h}{a} - 0.5\left(\frac{h}{a}\right)^3\right] \qquad \text{if } h < a \qquad (4.1a)$$

$$\gamma(h) = c \qquad \text{if } h \geq a \qquad (4.1b)$$

Exponential

$$\gamma(h) = c \cdot \left[1 - \exp\left(-\frac{h}{a}\right)\right] \qquad (4.2)$$

Gaussian

$$\gamma(h) = c \cdot \left[1 - \exp\left(-\frac{h^2}{a^2}\right)\right] \qquad (4.3)$$

Power

$$\gamma(h) = c \cdot h^a \qquad (4.4)$$

where the variogram, $\gamma(h)$, is defined by the equation:-

$$2\gamma(h) = E\{[Z(x) - Z(x + h)]^2\} \qquad (4.5a)$$

Here Z(x) is the value of a certain hydraulic property at location x, and h is the separation between two points at which the property is measured or defined. ("E" stands for "expected value".)

It is easily shown that $2\gamma(h)$ is thus equivalent to the variance of the difference between hydraulic property values pertaining to points a distance h apart taken over the area where the variogram is applicable; i.e.

$$2\gamma(h) = \mathrm{Var}[Z(x) - Z(x + h)] \tag{4.5b}$$

Variograms can be applied to native hydraulic properties, or to transformed property values (most often log-transformed property values). Furthermore, a variogram can be isotropic or anisotropic. In the latter case specification of the variogram requires that the direction of elongation of the ellipse of anisotropy be supplied. For the utilities described herein this can be oriented at an arbitrary angle to the finite difference grid.

Variograms can be "nested" to form a "geostatistical structure"; for the programs described herein, full characterisation of the geostatistical structure of an area can be based on the summation of up to 5 separate variograms plus a "nugget" (ie. a fully random component with no spatial correlation). Individual variograms within a nested structure can be of different types and of different anisotropy. In the groundwater context, however, it is rare that spatial variability will be characterised in enough detail to require more than one variogram for its specification.

Two types of kriging are permitted in the programs described herein, viz. "simple kriging" and "ordinary kriging". The difference between these is discussed in any geostatistical textbook. Often it is better to employ ordinary kriging when using pilot points to parameterise a model domain, for simple kriging requires that the mean value of the hydraulic property throughout the model domain be specified.

One of the advantages of using kriging as a method of spatial interpolation is that the factors by which property values at known points (ie. pilot points) are multiplied to obtain property values at unknown points are independent of the property values at the known points. Thus a set of "interpolation factors" can be calculated in advance, and re-applied to the property values assigned to pilot points as these values are updated. This makes the method ideal for use in the nonlinear parameter estimation context where property values at pilot points are continually updated as the parameter estimation process proceeds.

Another benefit of using kriging for spatial interpolation is that kriging is an "exact interpolator". This means that the interpolated property value at the location of each pilot point is equal to the property value assigned to that pilot point.

As a by-product of spatial interpolation using kriging, the uncertainty of interpolated property values can be calculated together with the interpolated property values themselves. This uncertainty estimate, too, is independent of the hydraulic property values assigned to the pilot points. Because this uncertainty can be calculated at all points throughout a model domain at which spatial interpolation is required (normally the centres of finite-difference cells), it can be visualised using the contouring and/or shading functionality available through a model array editor/viewer. Note, however, that calculation of uncertainty in this manner assumes minimal uncertainty associated with property values at pilot points. Where the latter are estimated as part of the model calibration process, this is certainly not the case. However such a contour map may be useful in judging where to add more pilot points if it is felt that extra points are warranted.

4.3 Parameter Estimation Using Pilot Points

When parameterisation of a model domain is based on pilot points, PEST can be used to estimate the hydraulic property value pertaining to each of the pilot points. These values are then spatially interpolated to the centres of pertinent active cells within the model domain using kriging based on one or more of the variograms discussed above. Thus in assigning values to pilot points, PEST is actually implicitly assigning hydraulic property values to part or all of the model domain.

There is no fundamental reason why kriging should be used for spatial interpolation between pilot point values rather than another method of spatial interpolation, for example inverse-power-of-distance, multiquadratic, splines, linear interpolation based on triangulation, etc. However there are some advantages to the kriging method that make it attractive for use in this role. Some of these have already been mentioned. Another one is that kriging tends to produce rather smooth, regular, interpolated property fields whereas some methods, such as splines, can produce “wavy” surfaces, especially if two pilot points are close together and are assigned very different property values. However, in this regard, kriging based on some variograms is better than kriging based on others. Experience has demonstrated that kriging based on the Gaussian variogram sometimes leads to spurious interpolated property fields. In general, kriging based on the exponential variogram seems to be the best behaved. Furthermore, it is often better to interpolate the logarithms of hydraulic properties assigned to pilot points rather than native hydraulic property values, for the resulting hydraulic property fields are often more regular and have a more realistic appearance. In fact the groundwater literature cites many examples where the hydraulic conductivity or transmissivity distribution within a study area is better characterised by a log variogram than a native variogram.

In spite of its advantages, care must always be taken when using kriging as a spatial interpolator. In some instances kriging factors can be negative, especially if using the Gaussian variogram. This can lead to interpolated properties which are greater than the highest property value assigned to any pilot point, or lower than the lowest property value assigned to any pilot point. Sometimes this can lead to spurious interpolated property values over parts of the model domain. To mitigate the adverse effects of such over- or under-estimation, the kriging utilities described herein allow the user to set maximum and minimum interpolated property values. A single value for each of these can be supplied, applicable to the entirety of the model domain, or maximum and minimum interpolation limits can be supplied on a cell-by-cell basis.

4.4 Variogram Selection

There are very few cases of groundwater model deployment where enough hydraulic properties have been measured at enough points to determine variograms which accurately describe the spatial distribution of these properties. In most cases this does not really matter, for in the present context variograms such as those described in Section 4.2 are more useful as an interpolation device than as an accurate representation of hydraulic property heterogeneity within a model domain. Furthermore, if hydraulic property values at pilot points are estimated as part of the model calibration process, it is not possible to determine a variogram in advance based on property values pertaining to these pilot points.

If kriging is based on a single variogram, rather than a nested variogram structure, the value of *c* in each of equations 4.1 to 4.4 does not affect hydraulic property values calculated through spatial interpolation of pilot point values (unless it is set too high when using the power variogram). However it does affect the kriging variance associated with interpolated values. Fortunately this is of little concern in the present context.

A little care must be taken in selection of an appropriate value for the parameter *a* in each of equations 4.1 to 4.4. In the first three of these equations *a* is related to the "range" of the variogram, this being the distance at which the variogram has almost reached its maximum value. In general, depending on the geometry of a study area and the number of pilot points used in the parameterisation process, a variogram range that is roughly equal to 2 to 3 times the average distance between pilot points is suitable for most occasions. The reason for this is more practical than scientific. If the range is shorter than the distance between most pilot points, then interpolated property values over much of the model domain will tend towards the average value of all points within the domain (or at least of those points within the search radius used for selection of points for interpolation to any particular cell centre). When pilot points are used in conjunction with PEST for flexible characterisation of spatial property distribution, this will hamper PEST's ability to impose any kind of hydraulic property "structure" over the model domain such as might be required if the model is to produce outputs which match field data well.

For the spherical variogram listed in Section 4.2 the range is approximately equal to *a*. for the exponential and Gaussian variograms the range is approximately equal to 3*a*. The power variogram has no range, for there is no distance at which the variogram reaches an upper limit. In this case the value for the variable *a* should be selected in such a manner that application of equation 4.4 yields a realistic estimate of the expected degree of spatial variability of a particular hydraulic property (or of its logarithm) over the model domain.

The value selected for anisotropy of a variogram can be critical in determining the nature of a spatially interpolated hydraulic property field. Anisotropy is generally defined as the ratio of the variogram range in the direction of maximum spatial uniformity to that in the direction of minimum spatial uniformity; thus it is generally greater than or equal to 1. As well as the magnitude of the anisotropy, the bearing of the direction of greatest uniformity must also be supplied as part of the geostatistical characterisation of an area on which kriging interpolation is based.

It should be noted that in using the utilities described herein, different variograms (with different properties) can be assigned to different zones within the model domain; thus these different zones can represent geological units with very different geostatistical structures.

It is not unreasonable to ask why the variables governing the shape of a variogram cannot be estimated together with the pilot point properties themselves as part of the nonlinear parameter estimation process. In theory, there is no reason why this cannot be done. However, in some instances there are definite disadvantages associated with this approach. The first is that, once the range has reached a certain value, model outputs tend to be insensitive to it (and, of course, if a geostatistical structure contains only one "nested" variogram they are completely insensitive to *c*). The second is that if the range is estimated at the same time as the pilot point property values, then kriging weights must be continually re-calculated. This can be a numerically-intensive process and can lengthen the time required for a PEST run considerably. However if the properties of the variogram remain invariant, then the same set of kriging factors can be used for every model run.

As mentioned above, ordinary kriging, rather than simple kriging, should be used on most occasions on which parameterisation of the model domain is effected through the use of pilot points. When simple kriging is used, kriging factors depend on the (externally-supplied) mean hydraulic property value over a model domain or zone. If the mean is estimated as part of the calibration process, all kriging factors need to be updated as the parameter estimation process progresses. This can be a time-consuming procedure.

Another advantage of using ordinary kriging rather than simple kriging in most instances of pilot point parameterisation is that, because the mean is not specified (or estimated) over a model domain or zone, it can effectively be allowed to vary spatially over this domain or zone by limiting the number of pilot points used in spatial interpolation to each cell centre on the basis of a "search radius", or by simply specifying the maximum number of points to be used in any one calculation of an interpolated value. The greater the search radius, the more homogeneous is a spatially interpolated property field likely to be. A smaller search radius can thus promote a greater degree of spatial variability; however it also limits the stability of the kriging process, and the effectiveness of any kriging-based regularisation employed in the parameter estimation process (see below).

In most cases where kriging is used in conjunction with pilot points for model parameterisation, a nugget value of zero should be used. However a non-zero nugget value can sometimes mitigate the effects of instability incurred through use of the Gaussian variogram. However, to be on the safe side, it is best to avoid the use of this type of variogram in the present context.

4.5 Locations of Pilot Points

When using pilot points for model parameterisation the user must decide where these pilot points should be placed. For some points the answer will be obvious; for example if hydraulic properties were measured in a bore (and if it is reasonable to assume that borehole-measured properties are suitable for use in a groundwater model), then a pilot point should be placed at the site of that bore. The property value assigned to that point may then be fixed during the parameter estimation process.

The locations of other pilot points may not be so easy to determine. However in most instances there are a number of simple criteria which can help in the selection of pilot point locations. One such criterion is that there should be a "good spread" of these points throughout the model domain or zone within which they are employed. This may necessitate that some points be placed at the boundaries of the model domain (often far removed from the sites of borehole head measurements or river flow measurements employed in the model calibration process). By placing some points near the boundaries, they can be used in conjunction with internal points to enforce some "structure" on the disposition of interpolated hydraulic properties throughout the model domain or zone. If such boundary points were absent, interpolated hydraulic property values may tend towards the mean of pilot point values as the boundary is approached, a situation that may prevent a good fit between model outputs and field observations from being achieved in certain hydraulic settings due to the constraints thus imposed on the hydraulic property interpolation process.

Another criterion that can be used to assist in the placement of pilot points is that they should not be placed too close together unless this is necessary for the model to obtain a good replication of disparate field measurements at nearby sites. As was mentioned above, if pilot points are placed too close together, instability of the kriging process can occur in some situations, particularly if using a Gaussian variogram. However, provided proper caution is exercised, it is often a good idea to place pilot points between bores in which there is a substantial head differential, in order that PEST can assign values to these points in a manner that is most beneficial for reproducing the head difference between these bores using the model. (Note, however, that the *calc_kriging_factors_auto_2d()* function of PLPROC partially overcomes problems associated with spatial variability of pilot point density.)

Another guiding principal is that, if possible, pilot points should be placed at locations at which key model predictions are most sensitive to their values. This can be particularly useful where calibration-constrained uncertainty analysis is undertaken in order to infer the effects of unknown, but plausible, geological heterogeneity on key model predictions.

4.6 Regularisation

One of the main advantages of using pilot points in preference to zones (or in combination with zones) is that the modeller does not need to define the "hydrogeological structure" of the model area ahead of the parameter estimation process. For example, in a certain study area it may be apparent from the shapes of piezometric contours, and from an inability to achieve a satisfactory fit between model outputs and borehole-measured heads based on a simple parameterisation of the model domain, that the subsurface is heterogeneous. However it may be very difficult to determine a zonation pattern that improves model-to-measurement fit to a desired level. In practice, a modeller will probably design a zonation pattern over the model domain based on intuition, geological insight, the shapes of piezometric contours and the results of previous calibration attempts, and will then use PEST to optimise the hydraulic properties assigned to these zones. If a satisfactory fit between model outcomes and field observations can be obtained for any particular zonation pattern, then this is taken as a necessary, but not sufficient, condition for acceptance of that zonation pattern. However, due to the inherent nonuniqueness of the inverse problem, there will almost certainly be other (possibly quite different) zonation patterns which will also do the job. But a modeller will rarely have the time, nor the patience, to search for too many of these alternative patterns.

If a modeller is unsure of the hydrogeological structure prevailing within a certain area, an alternative calibration methodology is to "let PEST find the structure" by calibrating the model on the basis of a superfluity of pilot points scattered liberally throughout the model domain. If the superfluity of pilot points is combined with regularisation information, and if PEST is asked to optimise parameters while working in regularisation mode, then its performance in optimising these parameters can often be quite spectacular. Furthermore, because of the superfluity of pilot points scattered throughout the model domain, PEST is able to establish the hydrogeological structure itself through the way in which it assigns property values to these points in order to reduce model-to-measurement misfit as much as possible. The problem of having to test different zonation patterns is thereby eliminated.

As is explained in PEST documentation, regularisation is the imposition of a set of relationships between parameters (or between functions of parameters), added to the calibration process as either observations or prior information. The addition of these relationships often brings uniqueness to the parameter estimation problem, even where there are far more parameters requiring estimation than it would be possible to accommodate on the basis of the observation data alone. If regularisation conditions are properly defined, parameterisation uniqueness (and hence numerical stability) results from the fact that these conditions on their own would be sufficient, or nearly sufficient, to parameterise the model domain, even if there were no data. (Of course, the addition of data to the calibration process results in a very different set of parameters than would be calculated if there were no such data.)

When used in regularisation mode (just as in parameter estimation mode) PEST is asked to ensure that the fit between model outcomes and field observations is as good as possible. However it is asked to do this in such a way that the departure between calibrated parameter values and an "ideal" parameterisation (as suggested by the regularisation observations or prior information) is as small as possible. Often these regularisation constraints take the form of a homogeneity condition, i.e. a constraint that differences between various pairs of parameter values (or their logs) be zero if possible. Thus, when used in regularisation mode with these constraints applied, PEST is asked to estimate a parameter set that deviates from uniformity only to the extent required to ensure a good fit between model outcomes and field data. Because the optimised parameter set is then defined not just in terms of its ability to allow the model to match field data well, but also in terms of its ability to replicate some "ideal" condition, solution to the inverse problem gains uniqueness. This results in stable

PEST behaviour and, if regularisation constraints are properly defined, a parameter set that is intuitively and geologically pleasing.

4.7 Predictive Uncertainty Analysis

The use of pilot points in model parameterisation can be very useful when exploring the uncertainties associated with predictions of subsurface contaminant transport. The fate of a contaminant is particularly sensitive to "geological fine detail". Unfortunately, it is rarely possible to resolve this fine detail during a calibration process, especially when this process is based only on head and flow measurements. Although the use of borehole contaminant concentration or travel time measurements can assist in this regard, rarely, if ever, will "parameterisation fine detail" be sufficiently well determined to prevent a large margin of error from being associated with predictions of contaminant fate. Hence, if computer modelling is to be used to maximum advantage in simulating contaminant transport processes under a particular site, the extent of uncertainty associated with contaminant fate predictions should be explored as part of the modelling process.

The use of pilot points to characterise the hydraulic property distribution of a model domain brings far more flexibility to the predictive uncertainty analysis process than the use of zones. One way in which predictive uncertainty analysis can be implemented is through direct predictive hypothesis-testing. Alternatively, ensembles of pilot-point based, calibration-constrained, random parameter fields can be obtained using the null space Monte Carlo methodology or the PESTPP-IES ensemble smoother. Direct predictive hypothesis-testing requires that a model be calibrated against historical data supplemented with an observation that a particular (often pessimistic) prediction related to contaminant fate actually occurs. Regardless of the method selected for uncertainty analysis, the use of pilot points (especially if there is a superfluity of these points) allows PEST to define hydrogeological structures within the model domain that allow the full range of contaminant fate scenarios to be explored.

4.8 Utility Programs

A full description of all of the programs available through the Groundwater Data Utilities for pilot-point-based parameterisation and ancillary regularisation and stochastic field generation is presented in Part B of this manual. These programs are tabulated below. See also PLPROC.

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.
GENREG	Inserts prior information pertaining to many different types of regularisation into an existing PEST control file.
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.
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.
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".
PPMDEF	Builds a parameter definition file for the use of ASENPROC, linking distributed parameters as employed by the adjoint state version of MODFLOW to pilot point parameters.
PPSAMP	Used in calibration-controlled Monte Carlo analysis. Samples stochastic fields at pilot point locations, interpolates between the pilot points and generates difference fields.
VERTREG	Adds "vertical regularisation" prior information to a PEST control file where parameterisation is based on pilot points.
ZONE2VAR1	Computes a parameter variogram where parameterization is based on a large number of zones of piecewise constancy, and is defined through a ZONMDEF output file. Assists in undertaking "variogram regularisation" as described by Johnson et al (2007).

ZONE2VAR2	Computes a parameter variogram much more quickly than ZONE2VAR1 because it employs the results of the parameter search process undertaken by the latter program as read from a binary file written by it.

Table 4.1 Programs used in pilot points parameterization of a groundwater model. See also PLPROC.

5. References

Deutsch, C.V. and Journel, A.G., 1998. GSLIB. Geostatistical Software Library and User's Guide. Oxford University Press, 2nd Edition, 1998.

Johnson, T.C., P.S. Routh, T.Clemo, W. Barrash and W.P. Clement. Incorporating geostatistical constraints in nonlinear inversion problems. Accepted for publication in Water Resources Research.