UCODE "EASY READ" INSTRUCTIONS

GENERAL (PG 24 OF UCODE MANUAL)

INPUT FILE CONSTRUCTION (PG 24)

The UCODE input files are described in detail in this section. As discussed in the previous section 'Filename Restrictions,' most of these files need to have names of the form *fn*.*, where fn is replaced by the prefix defined in the UCODE run command, and the suffix * is specified for different types of files. For example, the universal file suffix needs to be uni, and the universal file is referred to as *fn.uni* in this report. Italicized names are used to remind the reader that the *fn* prefix is replaced by the user-defined prefix from the run command.

The following sections describe the universal, prepare, template, function, and extract input files in detail and provide annotated examples of each file. Four characteristics are shared by all but the template files, so are listed here and briefly mentioned again below.

free format — Each line of the input files is read using free-format, but the **number of variables specified for each line needs to be adhered to strictly**.

blank lines — Blank lines are permitted anywhere in all files, except **for *fn.uni* in which no blank lines are permitted**.

— Comments are included in all of the files by inserting the symbol #, after which all characters are ignored by UCODE. In the universal and extract files, comments can be inserted as the first character of a line or to the right of input data after one or more blank spaces. In **the prepare file, comments need to occupy an entire line and the # needs to be located in the first column.**

END — The last line of the data input of each file needs to be **END** or **end**, starting in the first space. Anything after this line will not be read, which provides a convenient place for notes. On some computers at least one line needs to follow this line; that is, an "enter" or "return" needs to be inserted at the end of the line.

UNI (PG 25 of UCODE MANUAL)

Universal Input File—*fn.uni* (PG 25)

The universal input file, *fn.uni*, consists of solution control variables, the name of the inversion model, the command(s) needed to execute the application model(s), variables governing output, and a list of observation information.

Directions for the Universal Input File

Except for the last part of this input file, which contains the observation information, each line contains only one variable. Comments can be included either by placing a **# sign in the very first column of a line, or by placing a # sign after one or more blank spaces at the end of a data input line. Blank lines are not permitted before the END line. The data input, in order of appearance, are as follows.**

PHASE - Specifies the function UCODE performs. It is useful to begin with PHASE=1 and proceed to 2 and(or) 22, and then 3. Runs with PHASE=33, 44, and 45 generally are run only using a satisfactorily calibrated model. Phase 11 produces values which can be used to create a sum-of-squared, weighted residuals contour graph, and may never be used in some circumstances. The function of each PHASE is described briefly in the following table.

PHASE	Function
1	Parameter substitution and forward modeling using the starting parameter values specified in the prepare file.
11	Substitutes parameters, performs a forward model run, and calculates the sum-of- squared, weighted residuals objective function for many sets of parameter values. PHASE=11 produces data sets from which objective-function contour graphs can be produced. Execution of PHASE=11 requires modification of the search-string lines of the *fn.pre* file.
2	Sensitivities at starting parameter values
22	Sensitivities and parameter variances, covariances and correlations at starting parameter values. Execution time for 22 is about twice that of 2 because central differences, rather than forward differences are calculated.
3	Perform regression.
PHASE=33 and 44 need to be preceded by executing UCODE with PHASE=3 and GRAPH=1 in the same directory (GRAPH is a variable in *fn.uni*).	
33	Calculate the modified Beale's measure of model linearity using methods discussed by Cooley and Naff (1990) and Hill (1994).
44	Calculate predictions and their linear confidence and prediction intervals. Only PHASE is read from the universal file.[1]
PHASE=45 needs to be preceded by executing UCODE with PHASE=44 in the same directory.	
45	Calculate differences and their linear confidence and prediction intervals. Only PHASE is read from the universal file.[1]

1. Calculated using a slightly modified version of the computer program YCINT (Hill, 1994).

The following six variables control the sensitivity and regression calculations and define the inversion algorithm.

DIFFERENCING - Controls the method used to calculate sensitivities during the parameter-estimation iterations (starting with 1 is recommended):
1 forward differencing is used; and
2 central differencing is used and execution time is likely to double.

TOL When parameter values change less than this fractional amount between regression iterations, parameter estimation converges (0.01 is recommended).

SOSR When the sum-of-squared, weighted residuals changes less than this fractional amount over three regression iterations, parameter estimation converges. Ideally, for the final results convergence is achieved by satisfying the TOL criterion so that SOSR can equal 0.0 (in which case SOSR is not used as a convergence criteria). Values of SOSR of 0.01 and even 0.1 can be useful, however, in the early stages of model calibration because it stops the regression when it is not progressing.

NOPT Controls whether quasi-Newton updating will be used when the sum-of-squared, weighted residuals changes less than 0.01 over three regression iterations (using 0 is recommended):
0 no quasi-Newton updating; and
1 apply quasi-Newton update when criterion is met.
NOPT = 1 may facilitate convergence of highly nonlinear problems (Hill, 1998).

MAX-ITER - Maximum number of regression iterations (starting with twice the number of parameters is recommended; it is rare that more iterations will be helpful for problems that do not converge).

MAX-CHANGE - Maximum fractional change of a parameter value allowed in one regression iteration. For example, if MAX-CHANGE = 2.0, a parameter value of 1.0 will not be allowed to change by more than 2.0 (MAX-CHANGE times the parameter value). Consequently, the new value will be between -1.0 and 3.0. A parameter value of 2.0 will not be allowed to

change more than a value of 4.0 (again, MAX-CHANGE times the parameter value), and the new value will be between -2.0 and 6.0). This maximum change is applied to the physical parameter value, not its log transform. Exceptions are discussed in Hill (1998, Appendix B). (MAX-CHANGE = 2.0 is common, but smaller values may help an oscillating regression to converge.)

INVERSION ALGORITHM - Name of nonlinear regression executable (MRDRIVE, as distributed; generally the path name needs to be specified).

The following lines control the application model(s)

N-APPLICATIONS - Number of application models used.

APPLICATION MODEL EXECUTION COMMANDS - Commands needed to run the application models. Application models can be pre-processor or post-processors. If the application model requires screen input, create a batch file to run the application model. In many situations, the batch file will contain just one line with the application model command followed by < *filename*, where the file *filename* contains information that is typically input to the screen. The example in Appendix A illustrates a rather complex connection of files (**m.bat**, **in**, **ex.fil**, and **out**). The file **m.bat** runs the application model in batch mode, defining the file **in** as the standard input and the file **out** as the standard output using <**in** and >**out**, respectively. The file **in** simply contains the filename **ex.fil** , which would need to be typed in answer to a query on the screen if the application model were executed directly, as opposed to in batch mode. Specifying the file **out** results in the diversion of printing from the screen to **out**, thus allowing messages provided by UCODE concerning the regression to appear on a less cluttered screen.

The following four variables control output, and do not influence the solution.

SCALE-SENSITIVITIES - Controls the scaling applied to the printed sensitivities. (typically, 1, 2, or 3 are used, rarely is the user interested in viewing unscaled sensitivities):

0 No scaling is applied, and unscaled sensitivities are printed.

1 Dimensionless scaled sensitivities are printed. Sensitivities are scaled by the parameter value times the square-root of the weight, resulting in dimensionless numbers. Composite scaled sensitivities also are printed.

2 One-percent scaled sensitivities are printed. Sensitivities are scaled by the parameter value divided by 100, resulting in numbers with the dimensions of the observations.

3 Both dimensionless and one-percent scaled sensitivities are printed.

PRINT-INTERMEDIATE - Controls whether residuals and sensitivities are printed for intermediate iterations (using 0 is recommended):

0 no printing for intermediate iterations; and

1 printing for intermediate iterations.

GRAPH - When PHASE=3, controls printing of data for graphical evaluation of residuals and creation of input files needed for executing PHASE=33 and 44:

0 do not print post-processing files; and

1 print post-processing files (generally recommended because the files generally are not large and it is convenient to have them available).

NUMBER-RESIDUAL-SETS – The number of sets of residuals that will be produced and written to files *fn._rp, fn._rd,* and *fn._rg* for evaluation of apparent non-randomness of residuals. All sets used need to be generated in the same run, so if additional sets are desired, execute PHASE=3 again (using final parameter values as starting values to minimize execution time), specifying the total number of desired sets of residuals.

LIST OF OBSERVATIONS - A list of observations follows; one line for each observation. This observation list needs to be coordinated exactly with the extract input file (*fn.ext*) discussed later, in that the same OBS-NAMEs need to occur in both files. If a mismatch occurs, UCODE prints an error message and stops. It may be convenient, but is not necessary, to put the observations in the same order in the two files. To list them in the same order, use the instructions for the extract file to determine an order that

considers the extraction effort. Each line in the list of observations includes the following variables.

OBS-NAME OBS-VALUE STATISTIC STAT-FLAG PLOT-SYMBOL The five values need to appear on one line separated by one or more blanks, and need to be in this order. The variables are defined as follows.

OBS-NAME - Observation name including up to 12 non-blank characters. OBS-NAME need not occupy 12 spaces, but needs to be followed by a blank space.

OBS-VALUE - Observed value.

STATISTIC - Statistic used to calculate the weight for the observation. The statistic can be a variance, standard deviation, or coefficient of variation, depending on the value of STAT-FLAG. The three options are provided so that the user can specify the statistic that is most meaningful in a given situation. For all three options, the statistic is used to calculate the variance, and the weight for the observation is set to one divided by the variance. For further discussion about calculation of weights and suggestions for determining STATISTIC, see Hill (1998, 'Weights for Observations and Prior Information' and Guideline 6).

STAT-FLAG - A flag indicating whether STATISTIC is a variance, standard deviation, or coefficient of variation.

STAT-FLAG	STATISTIC
0	variance
1	standard deviation
2	coefficient of variation

PLOT-SYMBOL - Printed in files for graphical analyses to allow control of the symbols used when plotting data. Commonly, different values for PLOT-SYMBOL are used for different types of observations. The utility of this input value depends on the plotting program.

END - Indicates the end of the data input for this file.

PRE (PG 25 of UCODE MANUAL)

Prepare File—*fn.pre*, Template, and Function (*fn.fnc*) Files (PG 30)

Unlike the other input files, the prepare input file is associated with additional input files. These associated files are the template and function files, and also are discussed in this section.

Directions for the Prepare Input File

Indicator codes are used in the prepare file to indicate the purpose and composition of each line. Indicator codes are #, F or f, <, >, /, and P or p, and need to be in the very first column of all nonblank lines in the prepare file. Most lines can appear in any order, except as follows (these exceptions also are noted below): (1**) if the F (or f) indicator code is used, only the # indicator code can precede it**; (2) the **< and > indicator codes are associated pairs** with the < defining a template file and the following > specifying the associated application model input file; and (3**) all lines beginning with P need to follow any lines beginning with /**. Blank lines in the prepare file are ignored, and can be used to make the file easier to read. The indicator codes and information that needs to follow them are described here.

Comments - any information you may wish to note in the prepare file. Comments cannot be located as flexibly in the prepare file as in the universal, function, and extract files: in the prepare file the # indicator code only can be located in the first column of a line; they cannot follow input data on the same line.

F Indicates whether functions will be applied to parameters before substitution, and needs to be the first line in the prepare file that is not blank or a comment. Only two possibilities exist:

F yes Use the function file.

F no Do not use the function file.

If the indicator code **F** does not appear in the prepare file, no function file is used.

< The name of a template file which contains symbols for which values are to be substituted.

> The name of the application model input file that is to be created by substituting numbers into the preceding template file.

/ Each of these lines defines a parameter name, provides information for finding where the value of the parameter is to be substituted in a template file, specifies the starting parameter value, and lists other parameter information. Each line contains eight items separated by one or more blanks, and ordered as:

/SEARCH-STRING START-VALUE REASONABLE-MINIMUM REASONABLE-MAXIMUM PERTURBATION FORMAT LOG-TRANSFORM ESTIMATE

Each item is described below.

SEARCH-STRING - A parameter name preceded by an exclamation point, and followed by commas and a final exclamation point; for example, !Rch1,,,!. The parameter name is used to label parameter-related values in UCODE output files, to define prior information (see the P indicator code below), and to define parameter-substitution functions (see the section 'Parameter-Substitution function file construction' below). Parameter names can include any combination of upper and lower case letters (A-Z, a-z) and numerals (0-9). The total number of characters in SEARCH-STRING, including the exclamation points, defines the number of spaces used to insert numbers into template files to create application model input files. For example, SEARCH-STRING !Rch1,,,! provides inserted numbers with nine spaces. A single parameter can be used to replace numbers in more than one template file to produce more than one application model input file, in which case the SEARCH-STRING for the parameter will appear in more than one template file.

START-VALUE - The starting value for this parameter, with which the parameter-estimation iterations begin. Do not specify the log of log-transformed parameters (discussed below); they are calculated by the program.

REASONABLE-MINIMUM - For PHASE=3, this is the reasonable minimum value for this parameter. The value is used solely to determine how the final optimized value of this parameter compares to a reasonable range of values. Do not specify the log of log-transformed parameters; they are calculated by the program. For PHASE=11, this value is used as the lower limit of the range over which the parameter value is varied for calculating the sum-of-squared, weighted residual objective function.

REASONABLE-MAXIMUM - For PHASE=3, this is the reasonable maximum value for this parameter. This value is used solely to determine how the final optimized value of this parameter compares to a reasonable range of values. Specify native values even if the parameter is log-transformed. For PHASE=11, this value is used as the upper limit of the range over which the parameter value is varied for calculating the sum-of-squared, weighted residuals objective function.

PERTURBATION - For most PHASES, this is the fractional amount the parameter is perturbed to calculate sensitivities. Starting with 0.01, one percent of the parameter value, is recommended. A larger value is needed if too many calculated sensitivities for the parameter are printed as 0.0. If a string of asterisks or if a divide-by-zero-error occurs; PERTURBATION can be increased or, in some situations, FORMAT (see below) can be redefined to provide more significant digits. For PHASE=11, PERTURBATION is the number of values the parameter is assigned, bounded by the values specified as REASONABLE-

MINIMUM and REASONABLE-MAXIMUM. For PHASE=11, PERTURBATION needs to be greater than 1; for all other phases, values less than one are needed.

FORMAT - The format for the value of the SEARCH-STRING variable as read by its application model. This format needs to be compatible with the format specified for the input file in the application model documentation. (See the section 'Format Configuration Rules'). If a function is applied such that the value substituted for the SEARCH-STRING is many orders of magnitude different than the parameter value, the specified format needs to provide a format with sufficient significant digits to accommodate the computed value. UCODE supports C computer-language format configurations, as discussed in the following section.

LOG-TRANSFORM - Set this variable to 1 to log-transform the parameter and 0 otherwise. Note the restriction when applying prior information to log-transformed parameters below. Typically log-transformed parameters are those for which negative values are not reasonable. For PHASE=11, increments are generated in log space if the parameter is log transformed and both the native parameter value and its log transform are included in the output file for plotting.

ESTIMATE - For PHASE=3, set to 1 to estimate the parameter by including it in the regression, and to 0 to leave the parameter at the specified value throughout the regression. For PHASE=11, set to 1 to vary the parameter value using the specified values of REASONABLE-MINIMUM, REASONABLE-MAXIMUM, and PERTURBATION, set to 0 to leave the parameter value at START-VALUE.

P These statements define prior information on parameters. Search strings for all parameters must be specified before the prior equations are defined.

Each definition of prior information occupies a single line composed of the following variables:

P EQUATION stat STAT flag STAT-FLAG plot PLOT-SYMBOL

where **P** is the indicator code, **EQUATION** is defined below, **STAT**, **STAT-FLAG**, and **PLOT-SYMBOL** are variables for which numbers are read, and **stat**, **flag**, and **plot** are labels. Items following **EQUATION** need to be separated by the indicated labels (**stat**, **flag**, and **plot**); the labels and variables need to be separated by at least one,spaces. **EQUATION** and the variables are defined as follows.

EQUATION - An equation of the form:

PVALUE = C1 x P1 & C2 x P2 . . . . .,

where the components are defined as follows. Items within the equation can be separated by any number of spaces.

PVALUE - The prior information value. Specify the native value, even for a log-transformed parameter.

= Indicates that the simulated value to be compared with PVALUE is calculated as follows.

C1, C2 Coefficients with values as specified by the user.

x Indicates multiplication. Needs to be preceded and followed by one blank space.

P1, P2 Parameter names, such as Rch1, as defined within the SEARCH-STRINGs defined earlier in the prepare input file. For the prior information calculation, these names are replaced by the parameter value.

& - Indicates that the preceding and following products are to be summed; thus, it performs like a +.

For one parameter, an example equation is 120 = 1 x K1. For parameters with LOG-TRANSFORM = 0, the equation can contain as many products as desired. For example, if prior information indicates that

the annual recharge rate is 10 inches, and recharge is estimated for 6 months of winter and for 6 months of summer (using parameter names RCHWINT and RCHSUMR), then **EQUATION** would be:

10 = 0.5 x RCHWINT & 0.5 x RCHSUMR

A similar type of equation on prior information could be applied to hydraulic conductivity values of multiple units determined by an aquifer test for which a single effective hydraulic conductivity was estimated and is used as prior information. As noted next, the multiple hydraulic conductivity parameters involved could not be log-transformed.

For log-transformed parameters only one term can be used in the equation. Thus, for log-transformed parameters, **EQUATION** is of the form

pvalue = C1xP1

STAT - Statistic used to calculate the weight for the prior information. As for STATISTIC in the *fn.uni* file, the statistic can be a variance, standard deviation, or coefficient of variation, depending on the value of STAT-FLAG. For all three options, the statistic is used to calculate the variance, and the weight for the prior information is set to one divided by the variance. **For log-transformed parameters, specify the log-transformed statistic, even though PVALUE is a native value**.

STAT-FLAG -- A flag indicating whether STATISTIC is a variance, standard deviation, or coefficient of variation.

STAT-FLAG	STATISTIC
0	variance
1	standard deviation
2	coefficient of variation

PLOT-SYMBOL - An integer printed in the UCODE output files used for graphical analyses. Different values for plot-symbol can be used to indicate different types of observations so they can be differentiated with a unique symbol on a graph. The utility of

PLOT-SYMBOL will depend on the graphical software being used.

Completing one of the examples from above, a complete line beginning with the **P** indicator code is:

P 10 = 0.5 x RCHWINT & 0.5 x RCHSUMR stat 0.20 flag 2 plot 4

END - indicates the end of data input for this file

FORMAT Configuration Rules

To specify FORMAT for the values replacing the SEARCH-STRING, C language formatting rules are used in UCODE. The most commonly used formats are described here; the numbers in the examples can be changed as needed.

%6d Integer. Overall width is 6 places. 6 places are available for printing the value, and this needs to include a negative sign to allow for integers less than zero.

%10.4f Real number. Overall width is 10 places with 4 digits to the right of the decimal point. For example, -1496.4945 would require a format of %10.4f. If the significant digits to the left of the decimal point are too numerous for the number to be expressed within the defined overall width (10 in the example), many computers will print asterisks (*) instead of a number. For example, if %9.4f is specified to print the number -1496.4945, commonly nine asterisks are printed.

%10.3e Real number in exponential notation. Overall width is 10 places, with one significant digit to the left of the decimal point. The number of significant digits after the decimal point depends on whether the number is negative or positive. For positive numbers, three significant figures are included to the right of the decimal point so that there are four significant digits total. For negative numbers, two significant digits are included to the right of the decimal point so that there are three significant digits total. For example, a FORMAT of %10.3e produced the numbers 2.313e-002 and -

1.20e+000. If the exponential format is written more generally as %*a*.*b*e, *a* minus *b* needs to be at least 7, and the number of significant digits printed will be *b+1* for positive numbers and *b* for negative numbers.

TPL (PG 37 of UCODE MANUAL)

Template File Construction **(PG 37)**

Template files generally are most easily constructed by starting with a working application model input file and identifying the numbers for which values are to be substituted by UCODE. Then, replace these numbers with SEARCH-STRINGS defined in the prepare input file. Such substitutions generally are straightforward, as shown in the annotated example shown below, but can be tedious.

FNC (PG 37 of UCODE MANUAL)

Parameter-Substitution Function File Construction—fn.fnc **(PG 37)**

The function input file is an optional file which can be used to provide instructions for manipulating parameter values before substituting them into template files. If used, **the first line that is not blank or a comment in the prepare file needs to start with indicator code F (or f), followed by the word "yes", as follows:**

F yes

Omitting this statement, or including the statement

F no

causes parameter values to be substituted directly, without manipulation. In such cases, the parameter-substitution function file is not read.

In addition to comment lines, the function file includes groups of lines with a specific structure. Each group completely describes how one parameter value is to be manipulated before being substituted into one template file. **The required structure begins with two indicator code lines, the first with indicator code :file and the other with indicator code :key", followed by any number of lines with no indicator code.**

The indicator code of the first line identifies the template file into which the manipulated values are to be substituted.

:file One or more spaces and the name of a template file follows; for example, :file *bcf.tpl*.

The indicator code of the second line identifies the parameter.

:key One or more spaces and a parameter name follows; for example, **:key** *k2m*.

These two lines are followed by any number of lines called occurrence and function lines. Each of these lines contains two entries: occurrence and function. Each occurrence entry identifies occurrence(s), in the file specified by the preceding **:file** indicator code, of the SEARCH STRING containing the parameter name specified by the preceding **:key** indicator code. The identified occurrence(s) are to be replaced using the function entry. The following paragraphs describe the occurrence and function entries in detail.

The occurrence entry is an integer or list of integers, where 1 indicates the first occurrence of the parameter name in the file specified by the preceding **:file** line, 2 indicates the second occurrence of the parameter name in the file, and so on. The counting starts at the top of the template file and proceeds left to right across each line. If the same function is to be applied to a number of sequential occurrences of the parameter name, the occurrence entry can be composed of the first and last number separated by two periods. For example, 1..6 indicates that the function will be applied to occurrences 1 through 6. If the occurrences are not sequential they need to appear on different lines. The occurrences under each :file indicator code need to be listed in increasing order . For example, the sequence 1, 3, 5 on three consecutive lines is acceptable, while the sequence 1, 5, 3 is not. For occurrences that do not appear in the list, the parameter value is substituted without modification.

The function entry is separated from the occurrence entry by one or more spaces. The function expressions can use any of the AWK functions listed in table 2, and parentheses can be used to dictate the order of the operations. When parentheses are not used, common mathematical operation order is applied.

Table 1: AWK functions for the function file

AWK function	Description
+, -, *, /	Addition, subtraction, multiplication, & division
x ^ y	Returns x raised to the power y.
atan2(y, x)	Returns arctangent of y/x.
cos(x)	Returns cosine of x; x is in radians.
sin(x)	Returns sin of x; x is in radians.
exp(x)	Returns the exponential function of x, base e.
log(x)	Returns the natural logarithm of x.
sqrt(x)	Returns the square root of x.
int(x)	Returns the value of x truncated to an integer.

The value of the parameter listed in the preceding **:key** line is represented by $x in the AWK expressions. Other parameter values can be represented in the function as the parameter name bounded by exclamation points, such as !rch1!. If two or more parameters are always functionally related in the same way, generally only one of the

parameters should be estimated by regression to avoid extreme correlation between the parameters involved.

Numbers of any format can be used in the AWK expressions, including integers, real numbers, and real numbers expressed in exponential notation. Exponential notation is expressed using the common conventions; for example, e-8, E+10, and e-001.

The following is an example of a group of occurrence and function lines:

```
1          $x*2.0e-03
2..5        $x*35 + 6
6          $x*(!rch1!+!rch2!)
7          sin($x)
```

If manipulated values of the same parameter are to be substituted into more than one template file, another group of lines needs to be repeated for each template file. If the same template file includes more than one parameter for which the value needs to be manipulated before substitution, again another complete group of lines, including the **:file** line, needs to be repeated for each parameter.

EXT (PG 41 of UCODE MANUAL)

Extract File—*fn.ext* **(PG 41)**

The extract input file provides instructions for extracting values from application model output files. If needed, the extract file also provides instructions for using these values to calculate the simulated equivalent values that are to be compared with the observations specified in the universal input file.

Directions for the Extract Input File

Each line in the extract input file needs to be structured in one of the two ways: (1) the first space of the line needs to be one of nine indicator codes, or (2) the line needs to be blank or include only comments.

The NINE CODES ARE:

O < /…../ + - c i s P

In any line, anything after a # sign is ignored by the program, and this provides a convenient method of including comments in the file. The input items can appear in nearly any order, but are executed sequentially. Thus, file must be opened before moving forward a given number of lines, or extracting a value between specified columns. The symbol designating comments and the nine indicator codes are as follows.

Comments (can be placed anywhere; any subsequent text on the line is ignored by the program).

The first indicator code identifies an observation from the universal input file, and is called an observation indicator code.

o An OBS-NAME from the universal input file follows either directly or after one or more spaces.

A line with an **o** indicator code needs to be followed by instructions that define how to calculate a simulated value that is to be compared with the observation identified

by OBS-NAME using extracted values and user-defined factors. The remaining indicator codes perform the functions needed to accomplish these calculations.

The next indicator code identifies the application model output file from which the subsequent extractions occur. This is called the file indicator code.

< The name of an application model output file follows; for example, *<mod.out*. The same file can be opened any number of times, but each time directions need to be given starting at the top of the file. When a file is opened, the position is at line zero. In order to operate on the first line, a subsequent code of +1 (discussed below) must be specified.

Three indicator codes are used to locate a line containing one or more values to be extracted. These are called locating indicator codes.

/...../ A string of characters for which to search is listed between the forward slashes (the case of letters needs to be matched). For example, /HEADS/ would cause UCODE to search the file specified by the last file indicator code < for the next occurrence of the string 'HEADS'. Note: Long strings are more efficient.

+ Move down the following number of lines. For example, +3 indicates to move down 3 lines. To be positioned at the first line of a file requires that +1 be specified after opening the file.

- Move up the following number of lines. For example, -13 indicates to move up 13 lines.

To ensure consistent results, use location indicator codes that function correctly in the presence of normal output file variations. For example, the number of lines printed by solvers often varies from run to run, and consistent results can not be achieved using only the + and - indicator codes. Consistent results can be achieved by searching for a string that follows the solver output, and proceeding from there using + and - indicator codes.

Values are extracted with the extraction indicator code **c**.

c Extract the value that occurs between column numbers which follow the **c** and are separated by an underscore (_). Multiply the extracted value by a factor that equals 1.0 or a number which follows after a second underscore. For example, c52_63_0.5 extracts the value in columns 52

through 63 and multiplies it by 0.5; c52_63 extracts the value and multiplies it by 1.0. **NOTE:** Extract as many significant figures as possible, as discussed in the section "Calculation of Sensitivities by UCODE".

In simple situations, a line with observation indicator code **o** can be preceded or followed by a single line with file indicator code <, any number of lines with locating indicator codes, and a single line with extraction indicator code **c**. For example,

```
<mod.out      # open a file
o 1A          # OBS-NAME 1A
+1            # go to the first line in the file
c8_15         #  a value is extracted and a factor of 1.0 is used
```

Such a situation occurs when the product of one extracted value and its factor is directly comparable to an observed value specified in the universal input file, *fn.uni*.

For more complicated situations, UCODE includes three additional indicator codes, called function indicator codes, that define different ways in which extracted values and factors can be combined. These indicator codes may be followed by: one or more lines with file indicator code <, usually many lines with locating indicator codes, and usually many lines with extraction indicator code **c**. If all extractions are from the same file the single file indicator code can precede the function indicator code.

Two function indicator codes perform exactly the same mathematical function, in that they sum the products of extracted values and their factors. That is, they perform calculations of the form

$$Y = X1*F1 + X2*F2 + \ldots\ldots,$$

where

X1, X2, . . . are values extracted using a **c** indicator code;

F1, F2, . . . are the user-defined factors associated with X1, X2, . . . (the last of the three values listed after the **c** indicator code, or 1.0 if the third value is omitted); and

Y is a value that is to be compared with an observation.

Calculations of this form are commonly used both for interpolating values and summing values, and the different indicator code options are provided so that these different purposes can be identified more easily in the extract input file.

i Interpolation is performed. For example, **i**4 indicates interpolation of the following 4 extracted values. For interpolation, the user-defined factors commonly sum to 1.0.

s Summation is performed. For example, **s**20 indicates summation of the following 20 extracted values. In summation the user-defined factors commonly each equal 1.0.

The final indicator code performs a summation of the products of pairs of extracted values and their factors. That is, it performs calculations of the form:

$$Y = (X1*F1 * X2*F2) + (X3*F3 * X4*F4) \ldots,$$

where:

X1, X2, X3, X4, . . . are values extracted using a **c** indicator code;

F1, F2, F3, F4, . . . are the factors associated with X1, X2, . . .; and

Y is a value that is to be compared with an observation.

Calculations of this form are used in a variety of circumstances. A common ground-water example occurs when application model output files include flow rates and concentrations, and the observation of interest is a flux-averaged concentration. The applicable function indicator code is **p**, which stands for product.

p Sum a sequence of products of pairs of extracted values and their associated factors. For example, p2 indicates the product of the following 2 pairs of extracted values and their factors will be summed. That is, the product of the following 2 extracted values will be summed with the product of the subsequent 2 extracted values. Four values need to be extracted using four lines with extraction indicator codes (**c**). If the product of 5 pairs were to be summed, the code would be p5, and 10 extractions (code **c** lines) would follow before another observation name was specified.

END indicates the end of data input for this file.

GENERAL PHASE 44 & 45

(PG 47 of UCODE MANUAL)

Input Files for Predictions and Differences and Linear Confidence and Prediction Intervals (PG 47)

Predictions and differences, and their linear confidence and prediction intervals, are discussed in sub-section "Predictions and Differences and Their Linear Confidence and Prediction Intervals" within the section "Inverse Modeling Considerations". These quantities are calculated with UCODE using PHASE=44 and 45, with a slightly modified version of the computer program YCINT (Hill, 1994). In UCODE, PHASE=44 is used to calculate predictions for conditions that can (but may not) differ from the calibration conditions; PHASE=45 is used to calculate differences. The differences are calculated by subtracting values produced by a base simulation (simulated when PHASE=45) from values produced by a predictive simulation (simulated when PHASE=44). That is:

(value from predictive simulation with PHASE=44)

- (value from base simulation with PHASE=45)

= difference

Confidence and prediction intervals generally are calculated only for a satisfactorily calibrated model. PHASE=44 can be executed only after running PHASE=3 with GRAPH=1 in *fn.uni*, which creates files: *fn._tp*, *temp.u44*, *temp.p44*, temp.*f44*, and temp.*e44*. For PHASE=44, files *temp.u44, temp.p44*, *temp.f44*, and *temp.e44,* either need to be modified by the user, as described below and saved with the prefix *fn*; or, for those files where no modifications are needed, they must be copied to the filenames *fn.u44*, *fn.p44*, *fn.f44*, and *fn.e44, respectively*. The modified files need to be present in the directory in which UCODE with PHASE=3 was executed and from which UCODE with PHASE=44 is initiated. PHASE=45 can be executed only after running PHASE=44 in the same directory and modifying, then saving or copying the files: *temp.u45*, *temp.p45*, temp.*f45*, and temp.*e45*, to filenames: *fn.u45, fn.p45*, *fn.f44*, and *fn.e45*, respectively.

The rest of this section describes how to construct files for PHASE=44 and 45, including files *fn.u44*, *fn.p44*, *fn.f44*, and *fn.e44*, for PHASE=44, and files *fn.u45*, *fn.p45*, *fn.f45*, and *fn.e45*, for PHASE=45. These files are constructed in the same manner as the universal, prepare, function, and extract files discussed above, and are referred to as the prediction universal file, the prediction prepare file, and so on, and the base universal file, base prepare file, and so on. Although it may seem redundant to create these additional files, this organization makes it easy to return to PHASE=3 after executing the latter phases, as often is needed to evaluate alternative conceptual models.

PHASE 44 (PG 48 of UCODE MANUAL)

Directions for Input Files for Predictions--PHASE=44 **(PG 48)**

Input files for PHASE=44 are constructed as follows.

Copy or rename file temp.u44 to *fn*.u44, temp.e44 to *fn*.e44, temp.p44 to *fn*.p44, and, if used, temp.f44 to *fn*.f44.

1. Change the application model(s), the application model input files, and associated template files, as needed to simulate the **predictions**. The following checklist includes common changes, but additional changes also may be needed.
   a) Change the batch file(s) that execute the application model(s). If the name of the batch file is changed, or if a batch file is added, include this new information in the prediction universal file, *fn.u44*.
   b) Change the files used by the application model(s). For example, in a ground-water simulation, the imposed pumpage may need to be changed.
   c) Change the prediction prepare and function files, as required if application code input files used for the prediction simulation do not conform to the input files used during model calibration. In this circumstance, the prediction prepare file, *fn.p44*, needs to be changed to specify the file names pertinent to the predictive simulation. If used, the prediction function file, *fn.f44*, also needs to have file names changed, and, though unlikely, the occurrences also may need to be changed. If such changes are made, it is suggested that the application model(s) be checked to ensure that the substitutions produce the same system characteristics as they did for the calibration.
   d) As needed, change the batch files used to run the application model(s) for the prediction conditions.
2. Set PHASE=44 in file *fn.uni*. Although the rest of *fn.uni* is ignored for PHASE=44, usually it is most convenient not to modify the rest of the file.
3. List the predictions and specify information about the predictions using the prediction universal file, *fn.u44*. Use the last part of the file, which was used to specify observations in the *fn.uni* file. The format described for observations also is used for predictions, so that the lines are of the form:

OBS-NAME OBS-VALUE STATISTIC STAT-FLAG PLOT-SYMBOL

where, for PHASE=45,

OBS-NAME - Prediction name (can be up to 12 characters long).

OBS-VALUE - Specify a number, but the number is not used.

STATISTIC - The standard deviation or variance of the measurement error expected for the prediction used to calculate prediction intervals (see Hill, 1994, eq. 15, 16, and 17).

STAT-FLAG - Set to 0 if STATISTIC is a variance and 1 if STATISTIC is a standard deviation.

PLOT_SYMBOL - Used as before.

4. Edit prediction extract file *fn.e44*, which was produced by UCODE with PHASE=3 and GRAPH=1. As produced, *fn.e44* simply contains comment statements with instructions. The completed *fn.e44* needs to extract values from application model output file(s) using the same mechanisms defined for the extract file above, and define how to calculate the predictions from the extracted values.

PHASE 45 (PG 49 of UCODE MANUAL)

Directions for Input Files for Differences--PHASE=45 **(PG 49)**

Input files for PHASE=45 are constructed as follows.

Copy or rename file temp.u45 to *fn*.u45, temp.e45 to *fn*.e45, temp.p45 to *fn*.p45, and, if used, temp.f45 to *fn*.f45.

1. Change the application model(s) and application model input files as needed to represent the **base** conditions, which are used to calculate differences from the PHASE=44 execution. See the list presented for PHASE=44 for suggested changes. Set PHASE=45 in file *fn.uni*. Although the rest of *fn.uni* is ignored for PHASE=45, usually it is most convenient not to modify the rest of the file.
2. List the base quantities and specify information about them using the base universal file, *fn.u45*. Use the last part of the file, which was used to specify the predictions in *fn.u44*. The order needs to be identical to that of the predictions listed in *fn.u44* because the base values are subtracted from the predictions using that order. Thus, the first base value defined in *fn.e45* is subtracted from the first prediction specified in *fn.e44*, the second base value is subtracted from the second prediction, and so on. As before, lines in the last part of the file need to be of the form:

OBS-NAME OBS-VALUE STATISTIC STAT-FLAG PLOT-SYMBOL

where, for PHASE=45,

OBS-NAME - Name of the difference (can be up to 12 characters long).

OBS-VALUE - Specify a number, but the number is not used.

STATISTIC - The standard deviation or variance of the measurement error that would be likely if the base value were measured. STATISTIC is used to calculate prediction intervals (see discussion above).

STAT-FLAG - Set to 0 if STATISTIC is a variance, and 1 if STATISTIC is a standard deviation.

PLOT_SYMBOL - Used as before.

3. Edit the base extraction file *fn.e45*, which was produced by UCODE with PHASE=44. As produced, *fn.e45* is identical to *fn.e44*. The completed *fn.e45* file needs to extract values from application model output files that reflect the base case,

using the same mechanisms used in the extract file discussed above, and to define how to calculate the base values from the extracted values. The base values can be extracted in any order; they are subtracted from values produced when PHASE=44 using the order established in the *fn.u44* and *fn.u45* files.