UCODE with LUCIA step_by_step instructions

Prepared by Carsten Marohn

UCODE is a parameter estimation software for universal sensitivity analysis, calibration and uncertainty evaluation.

The software can be downloaded here:

https://geology.mines.edu/igwmc/ucode/

Corresponding citation:

Poeter, Eileen P., Mary C. Hill, Dan Lu, Claire R. Tiedeman, and Steffen Mehl, 2014, UCODE_2014, with new capabilities to define parameters unique to predictions, calculate weights using simulated values, estimate parameters with SVD, evaluate uncertainty with MCMC, and More: Integrated Groundwater Modeling Center Report Number GWMI 2014-02.

Running Ucode

1 Simple example Ucode 2014 distribution: Dupuit model

Files:

1. *dupuit.bat*: Calls *dupuit.exe*, the model that is calibrated here
2. *_ucode_dupuit.bat* calls *ucode_dupuit.in* and *ucode_dupuit*, *ucode.exe* and additional modules for residual analysis
3. *ucode_dupuit.in* is the main input file

   BEGIN OBSERVATION_DATA TABLE

   NROW=7 NCOL=5 COLUMNLABELS

   BEGIN OBSERVATION_DATA TABLE

   NROW=74 NCOL=5 COLUMNLABELS Obsname obsvalue statistic statflag GroupName

   END OBSERVATION_DATA
4. *dupuit.in.tpl* is an input template, i.e. parameterisation file, that includes the parameters to be modified:

   jtf !   # j template file is the delimiter that marks the parameters to be varied by Ucode

   5.1 2.7 50.   # These are fixed input parameter values

```
!K              ! !w              ! # UCODE will vary values for K and W (between !!)
```

5. *ucode_dupuit.ins*; instruction file, tells Ucode where to find model outputs. jif@ is the delimiter that includes the headers of the table (e.g. @ x head @), where the oputputs are found: Model output file *dupuit.out* shows this:

```
   x        head
    5.00000       5.49911
   10.00000       5.75451
```

2 LUCIA

2.1 File names and functions for LUCIA

1. ucode_lucia.bat runs ucode and points to the main input file

```
..\..\bin\ucode_2014.exe   ucode_lucia.in  ucode_lucia
```

2. ucode_lucia.in is the main input instruction file, where the type of statistics, iterations, tolerance for convergence etc. and the names of the input files are defined
3. *.tpl files build on the lut files; tables that are modified need to be entered here in text format. The parameters that are changed are entered with unique identifiers between @@, for the rest the normal parameter values are entered.
4. *.ins file tells Ucode where to find outputs in LUCIA output files.
5. *.obs files are the test point time series as arrays or ASCII files; the files contain observation values and observation names.
6. _lucia.bat runs lucia.py and convert_output.py
7. convert_outputs.py extracts data from lucia output tss files and converts them into data that are read in the *.obs files.

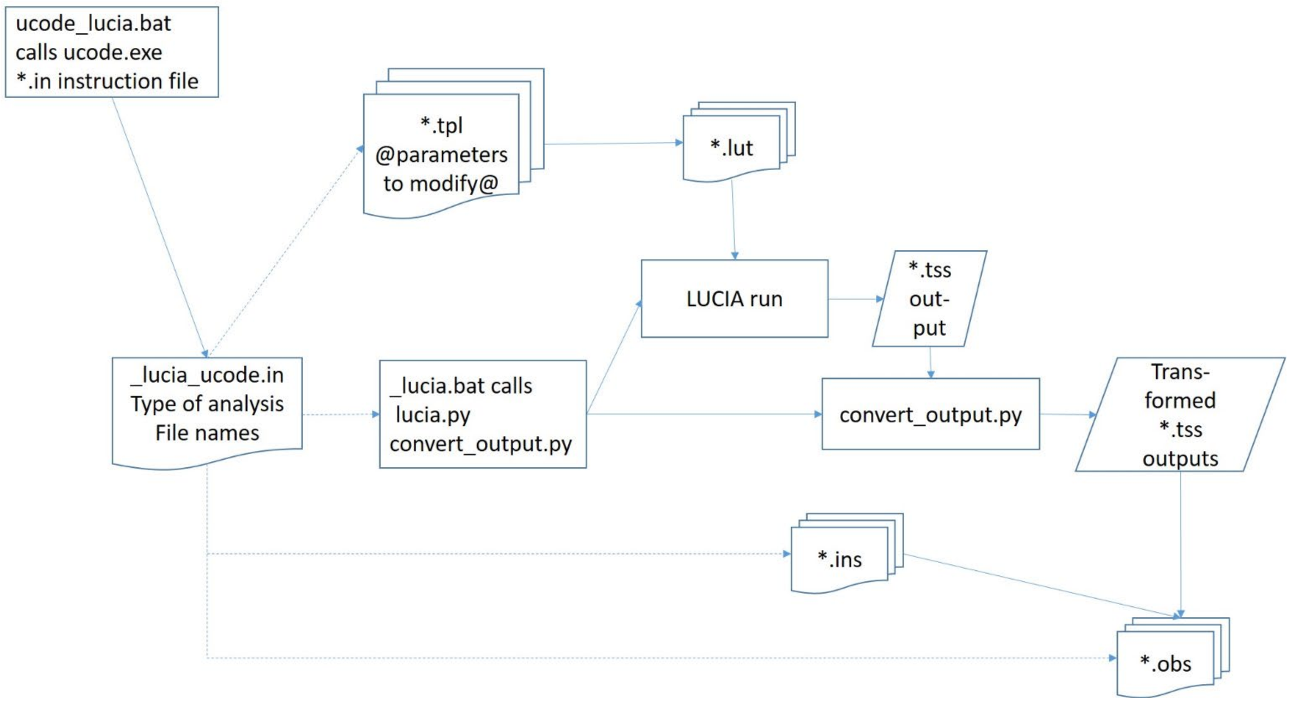
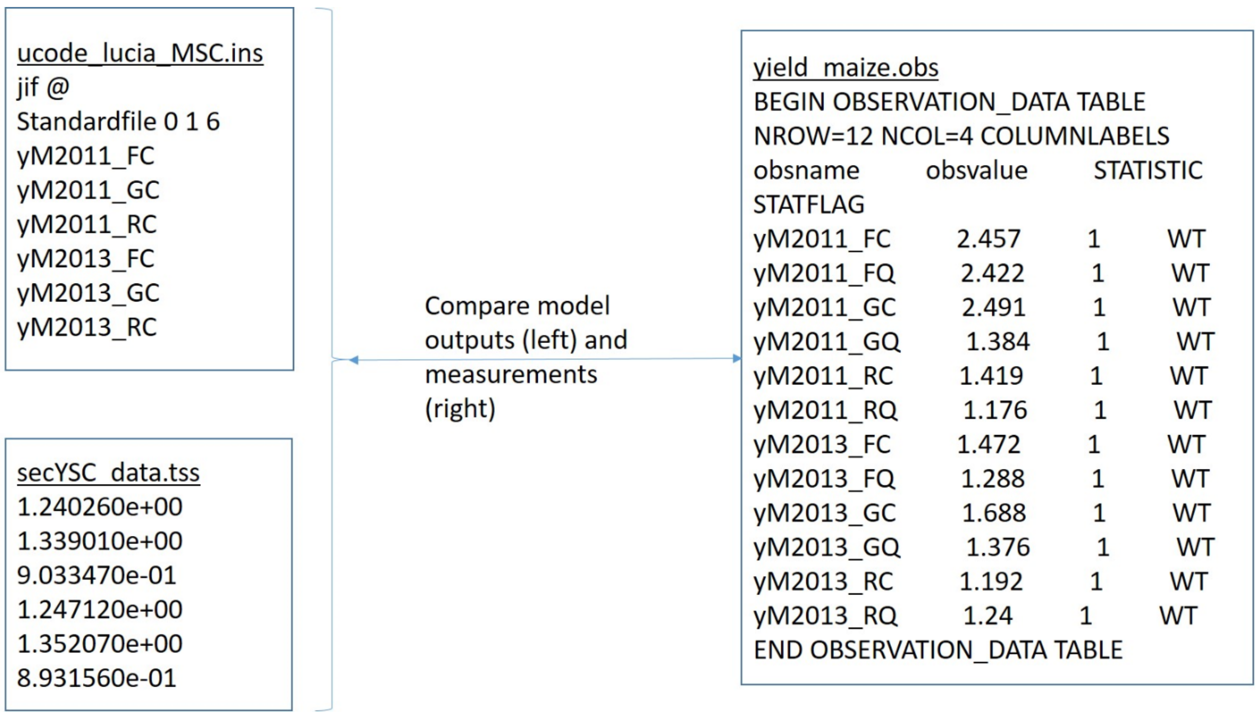

ucode_lucia.bat calls ucode.exe *.in instruction file
*.tpl @parameters to modify@
*.lut
LUCIA run
*.tss out-put
_lucia_ucode.in Type of analysis File names
_lucia.bat calls lucia.py convert_output.py
convert_output.py
Trans-formed *.tss outputs
*.ins
*.obs

Figure 1: Flow chart of a Ucode and Lucia run



```
ucode_lucia_MSC.ins
jif @
Standardfile 0 1 6
yM2011_FC
yM2011_GC
yM2011_RC
yM2013_FC
yM2013_GC
yM2013_RC
```

```
secYSC_data.tss
1.240260e+00
1.339010e+00
9.033470e-01
1.247120e+00
1.352070e+00
8.931560e-01
```

```
yield_maize.obs
BEGIN OBSERVATION_DATA TABLE
NROW=12 NCOL=4 COLUMNLABELS
obsname       obsvalue       STATISTIC
STATFLAG
yM2011_FC       2.457       1       WT
yM2011_FQ       2.422       1       WT
yM2011_GC       2.491       1       WT
yM2011_GQ       1.384       1       WT
yM2011_RC       1.419       1       WT
yM2011_RQ       1.176       1       WT
yM2013_FC       1.472       1       WT
yM2013_FQ       1.288       1       WT
yM2013_GC       1.688       1       WT
yM2013_GQ       1.376       1       WT
yM2013_RC       1.192       1       WT
yM2013_RQ       1.24        1       WT
END OBSERVATION_DATA TABLE
```

Figure 2: Details on the files that contain the process model outputs and observations

2.2 *The convert_outputs.py file*

This file resides in the /maps folder of the respective Lucia scenario.

It needs Lucia *.tss outputs files as inputs.

The output tss files (out_file_1 etc) and number of test points (TP_number) need to be entered. For each test point, an identifier and the columns and rows (representing points and timesteps) are specified by the user (yM2011_FC = yCSC_data.item((275, 1))).

The collected test point data are transformed and saved as a text file (np.savetxt("yCSC_data.tss", ...).

3 Parameters in the Ucode input file

UCODE_2005 requires a main input file, at least one template file, and at least one instruction file. The template files are used to define parameters to be estimated; the instruction files describe how to extract values from process-model output files. The main input file can read data from other files to facilitate data management.

3.1 *Syntax*

Block labels: Input in square brackets is optional. All input is case-insensitive and space-delimited. Sequence of block labels matters, not case-sensitive. Unnecessary blocks are ignored, so that they can be kept in a template of the input file. But: Misspelled block labels → default data are used without error message display. Check main output file (e.g. *ucode_lucia.#umcmc*).

If Table 4, Parameter_Values is defined as TABLE, then Parameter_data is overwritten.

Block format: Default is KEYWORDS, but TABLE, FILES etc are also possible.

Keyword format: keyword=value

For TABLES, the format is:

NROW=nr NCOL=nc [COLUMNLABELS] [DATAFILES=nfiles] [GROUPNAME=gpname]

If COLUMNLABELS are not defined, order of columns matters. If specified, COLUMNLABELS need to appear in the respective file / table.

If DATAFILES is on, data are read from file and the path has to be indicated (Table 6). Only file name (and path) needs to be indicated here, number of columns of a table or other is specified inside the file.

If SKIP=nskip is specified, nskip lines at the beginning of the file are ignored, and reading of data starts on the following line[1].

Further files:

Sensitivity files (more accurate calculations based on sensitivity equations possible than with perturbations).

Location files: Format *.xyzt. These are for spatio-temporal coordinates of observations for later plotting. They do not affect calculations.

Omissions of missing values: *.omit determines the mv

3.2 Input blocks

3.2.1 Options (optional)

Verbose

Derivatives_Interface: Data from sensitivity analysis of the process model can be read, code is Derivatives_Interface = "*.derint". Only one file allowed.

PathToMergedFile: Input data can be merged into one file (e.g. several derivatives files).

3.2.2 UCODE_Control_Data (optional)

Variables can be read in any order. Included variables generally are read using the default keyword format. TABLE format requires COLUMNLABELS because there is no default order for these variables.

ModelName and Units are such inputs, they are not mandatory.

Sensitivities: Calculate sensitivities or read them from a derivatives interface file listed in the Options input block

Optimize: Parameter estimation. Default output file *.uout

Linearity: execute the test-model-linearity mode and produce the data-exchange. Output: *.umodlin. The test-model-linearity mode needs to be run in a directory containing data-exchange files from a successful parameter-estimation mode run. Designate the same filename prefix on the command line that was used for the other runs.

LinearityAdv: ‘conf’ or ‘pred’: execute the advanced-model-linearity mode and produce files needed by

[1]Useful if dynamic (*.out files, n of LU lines omitted)

the program MODEL_LINEARITY_ADV. Output file umodlinadv_*, where * is 'conf' or 'pred'

Prediction: Determine predicted values and their sensitivities using the Prediction input blocks. The output is often used by postprocessor LINEAR_UNCERTAINTY. This may require use of alternative process-model input files and associated template files.

NonlinearIntervals: calculate nonlinear confidence intervals

SOSSurface: calculate sum-of-squared weighted residuals objective function values for the sets of parameter values defined. Main output file is fn.#usos and computed values are printed to fn._sos. Two options are yes and file: (a) The sets of parameter values are controlled by the keywords **Adjustable**, **SOSIncrement**, **LowerConstraint**, and **UpperConstraint** defined in the PARAMETER_GROUPS or PARAMETER_DATA input block, as described in the PARAMETER_DATA input block instructions. The number of sets (and the number of runs) equals the product of SOSIncrement for each parameter with **Adjustable**=yes, and can become large. Adjustable = no excludes parameters from being varied.

(b) The sets of parameter values are read from SOSFile. The parameters listed need to have adjustable=yes in the PARAMETER_DATA input block; not all adjustable parameters need be listed. If a listed parameter is not adjustable an error message is printed and execution stops.
If these are all "no" by designation or default, the forward mode is executed. For model calibration, the forward mode is typically performed first to check the execution of the process model(s) from the command lines, substitution of parameter values, and reading of model output values. SOSFile gives the path to the output file of this operation.

StdErrOne calculate statistics without using the value of the calculated standard error. Instead, use a value of 1.0.

EigenValues calculates and prints eigenvalues and their associated eigenvectors to the main output file

See Table 3 for the UCODE_2005 modes produced when these keywords equal "yes".

Print keywords default = yes are: **StartRes**, **intermedRes** and **FinalRes**. Same for sensitivity parameters: **StartSens**, **IntermedSens**, **FinalSens**. Options are:

DataExchange generates the data-exchange files containing data for graphical and numerical analysis (default = yes)

CreateInitFiles provides init files for sensitivity analysis.

3.2.3 Reg_GN_Controls (optional)

Modified Gauss-Newton regression method for estimating parameter values. Convergence criteria:

tolpar: Fractional tolerance based on parameter values.

Default=10^-2, which is 1% of the initial parameter value.

Set parameter-specific tolpar in the Parameter_Data input block.

If the parameter value equals 0.0, a value of 1.0 is used.

maxiter: Maximum number of parameter-estimation iterations allowed before stopping. Default=5.

TolSOSC: Parameter-estimation iterations stop if the fractional decline in the sum-of-squared weighted residuals over three parameter-estimation iterations is less than TolSOSC. If TolSOSC=0.0, it is not used. Default=0.0.

MaxChange: Maximum fractional amount parameter values are allowed to change between parameter-estimation iterations. Generic for all parameters unless specifically defined in a parameter block. Default=2.0, which means that parameter values can change as much as 200 percent.

MaxChangeRealm: Native, MaxChange applies in native space. Regression: MaxChange applies in regression space. In regression space MaxChange applies to log-transformed values for log-transformed parameters. Default=Native.

Marquardt parameter (used to improve ill-posed regression problems) related keywords: **MqrtDirection, MqrtFactor, MqrtIncrement**

QuasiNewton updating occasionally produces convergence for difficult problems. Options: **Qniter**, **Qnsosr**.

OmitDefault: When a simulated equivalent equals one of these numbers, the observation is omitted from the regression and a message is printed in its place in the UCODE output file. File needs to be located in the directory where UCODE_2005 is executed. No blank lines.

Stats_On_Nonconverge

OmitInsensitive, dazu gehören **MinimumSensRatio** und **ReincludeSensRatio**

Weighting of observations: If any observation is assigned a WtOSConstant > 0 in the Observation_Data input block, its weight can be calculated using simulated values and **TolParWtOS** is used. TolParWtOS×TolPar equals the parameter-change threshold below which simulated values are used to calculate weights on observations with WtOSConstant>0. Above this threshold, observed values are used to calculate the weights. Default=10.

Modification of the **trust region** with the double dogleg strategy to solve difficult problems and reduce number of iterations. Dogleg or hookstep as variations. Default=no.

MaxStep: If the regression is moving too slowly, assign MaxStep a value that is larger than the default. If the regression is too erratic, assign MaxStep a value that is smaller than the default. Use of the maximum allowable step size (MaxStep) by the regression is taken to indicate that either (1) the problem is very poorly defined so that extreme and probably unrealistic parameter values are being estimated or (2) MaxStep is too small and impeding progress of the regression.

ConsecMax: Maximum number of times that MaxStep is used consecutively before execution stops. Default=5.

Scaling: Option to improve convergence.

3.2.4 Model_Command_Lines block (required)

Command to execute the process model (command, path or batch file) has to be specified first. CommandID does not have any influence.

Purpose can be **forward** (model run induced), **derivatives** (sensitivities) or **forward&der** for both.

3.3 Parameter input

3.3.1 Parameter_Groups (optional)

Groups make sense if they share common values. Default=ParamDefault. Any keyword from Parameter_Data block can also be used here. Groupname must be first on a line.

```
BEGIN PARAMETER_GROUPS KEYWORDS
groupname=Default adjustable=yes tolpar=0.005
END PARAMETER_GROUPS
```

For TABLE format CLOLUMNHEADERS must be used, because no default order exists.

3.3.2 Parameter_Data (required)

Overwrites Parameter_groups values. **ParamName** up to 12 letters, first one must be a letter, remaining can be numbers or: &@#:._

If blockformat is keyword, then ParamName must be first in every line and the others follow directly. If block format = TABLE without COLUMNHEADERS, then sequence matters. Default Column Order: PARAMNAME GROUPNAME STARTVALUE LOWERVALUE UPPERVALUE CONSTRAIN LOWERCONSTRAINT UPPERCONSTRAINT ADJUSTABLE PERTURBAMT TRANSFORM TOLPAR MAXCHANGE SENMETHOD SCALEPVAL SOSINCREMENT NONLINEARINTERVAL

StartValue is a huge number like 10^{38} (default). **LowerValue** and **UpperValue** should be reasonable (default 10^{38}). Estimated parameter values outside the range will be reported as they may indicate model errors.

Constrain (default=no) means that estimates outside the range are discarded. This can prevent numerical overflow but also disguise model error. Only possible when *Optimize=yes*. For *Optimize=yes*, *Constrain=yes*, the parameter must remain between **LowerConstraint** and **UpperConstraint**. For SOSSurface, SOSIncrement must also be determined.

Adjustable defines whether the parameter can be varied or not (default = no!).

PerturbAmt is the fractional amount of parameter value to perturb to calculate sensitivity. Commonly 0.01 to 0.10. Default=0.01.

Transform of all parameter values to log10. Weighted residuals are transformed to natural log. default=no.

TolPar and **MaxChange** as in Gauss-Newton section.

SenMethod defines how sensitivities are obtained (from the process model or Ucode, and by which method):

ScalePval is a minimum cut-off for numbers in scaled sensitivities that have to remain positive.

SOSIncrement only used with *SOSsurface=yes* and *adjustable=yes*. Equidistant intervals between upper and lower constraints, if values are log-transformed, then they are equidistant in log-space. **NonLinearInterval** calculates nonlinear intervals (?). Not included in TABLE default order, can only be used with column headers.

3.3.3 Parameter_Values (optional)

These can be used as alternative inputs, while the originals are preserved in parameter_data. E.g. if values are taken from previous Ucode runs, from the process model or from elsewhere. Required are **ParamName** and **StartValue**.

3.3.4 Derived_Parameters (optional)

Parameters may be required by the process model in a different unit than given by the crude parameter_data; these can be derived here. Requires **DerParName and DerParEqn** (just the equation right-hand side). Example:

```
BEGIN Derived_Parameters TABLE
nrow=1 ncol=2 columnlabels
derparname derpareqn
```

K T/b

END Derived_Parameters

This functionality can be used to link parameters together, that are closely correlated. One parameter can then be defined as a function of another one. See manual 2008.

3.4 Input of observations and predictions

Observation blocks must be omitted when running Ucode in prediction mode

3.4.1 Observation_Groups (optional)

GroupName, **UseFlag**, **PlotSymbol** (for plotting in other softwares), **WtMultiplier** for the entire group.

CovMatrix gives the name of the error variance-covariance matrix. A CovMatrix is needed to represent correlation between errors in the members of a group. One matrix is used to define the weighting for all the members in the group. Members with independent errors have zero off-diagonal terms in the matrix.

3.4.2 Observation_Data (required)

Important: ObsName, ObsValue, StatFlag are mandatory.

ObsName (unique), **ObsValue**, **StatFlag**:

GroupName, **Equation** may be specified, too. **NonDetect** determines whether values below detection level should be omitted or set to detection threshold. **WtOSConstant** see Gauss-Newton section. Default Column Order:

OBSNAME OBSVALUE STATISTIC STATFLAG GROUPNAME EQUATION

Example for observations written in a file:

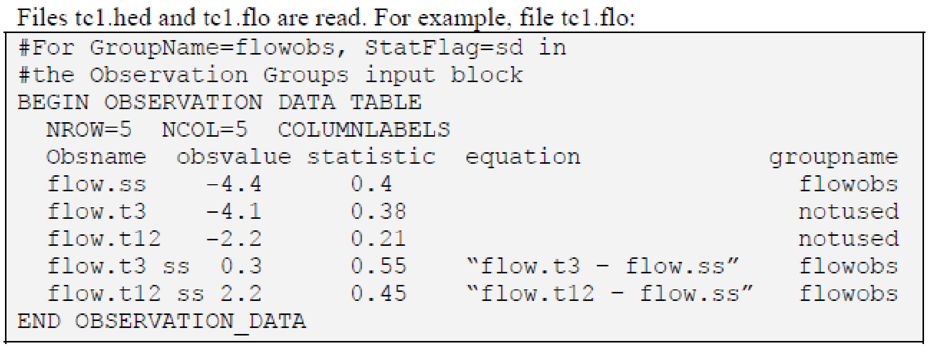
BEGIN OBSERVATION_DATA FILES

tc1.hed

tc1.flo

END OBSERVATION_DATA

Example for contents of the oberservation file:



Files tc1.hed and tc1.flo are read. For example, file tc1.flo:

```
#For GroupName=flowobs, StatFlag=sd in
#the Observation Groups input block
BEGIN OBSERVATION DATA TABLE
  NROW=5  NCOL=5  COLUMNLABELS
  Obsname  obsvalue statistic  equation              groupname
  flow.ss    -4.4      0.4                              flowobs
  flow.t3    -4.1      0.38                             notused
  flow.t12   -2.2      0.21                             notused
  flow.t3 ss  0.3      0.55    "flow.t3 - flow.ss"      flowobs
  flow.t12 ss 2.2      0.45    "flow.t12 - flow.ss"     flowobs
END OBSERVATION_DATA
```

Further options: Equations can be added (e.g. average of *hi1.0* and *hi1.0a* in *hi1.o_der*)

3.4.3 Derived_Observations (optional)

As before, here an example:

3.4.4 Prediction_Groups (optional)

GroupName, **UseFlag** (report and analyze the predictions in this group), **PlotSymbol** (for post-processing). Can be keyword or table (COLUMNHEADERS needed) input.

3.4.5 Prediction_Data (required)

PredName

RefValue: Reference value to which the prediction is compared. This value is used to calculate scaled sensitivities. If RefValue equals zero, the scaled sensitivities are set to zero.

MeasStatistic: A statistic used to calculate the variance with which the predicted quantity could be measured. The variance is printed in the _pv data-exchange file, which is used by LINEAR_UNCERTAINTY and the nonlinear-uncertainty mode of UCODE_2005 to calculate prediction intervals.

MeasStatFlag

MeasStatFlag	**Variance is calculated as**
VAR	MeasStatistic
SD	(MeasStatistic)

GroupName as before

Equation (from other PredNames). A prediction derived with an equation can not be used in conjunction with sensitivities calculated by the process model (SenMethod=-1 or 0 in the Parameter_Data input block). Rather, sensitivities need to be calculated by perturbation (SenMethod=1 or 2).

Default Column Order:

PREDNAME REFVALUE MEASSTATISTIC MEASSTATFLAG GROUPNAME EQUATION

BEGIN PREDICTION_DATA FILES

tc1.hed

tc1.flo

END PREDICTION_DATA

Example of file contents:

3.4.6 Derived_Predictions (optional)

From

Omitted Chapters on direct input of measurements and weighting

3.5 Ucode input file

3.5.1 Model_Input_Files Input Block (required)

ModInFile needs to be the first variable in keyword mode. It is the name of the process model's input file. **TemplateFile** is the name of the file Ucode produces that goes into the process model (for LUCIA, the lut file).

Template file construction:

The jtf or substitution delimiter for template files marks where the Ucode values are entered. Example:

Important: The field between two delimiters contains the parameter name and a large number of spaces. The field needs to be large enough to print the value with all required digits, so that accuracy is not affected.

3.5.2 Model_Output_Files Input Block (required)

ModOutFile, InstructionFile (*.in)

Category can be obs or pred. For Obs, process model outputs are treated as observations (and ignired if prediction=yes in the UCODE_Control_Data input block). For Pred model outputs serve as predictions in prediction mode.

Example in keyword format:

```
BEGIN MODEL_OUTPUT_FILES Keywords
modoutfile=tc1-fwd._os instructionfile=tc1.ins category=obs
END MODEL_OUTPUT_FILES
```

In table format:

```
BEGIN MODEL_OUTPUT_FILES TABLE
nrow=2 ncol=3 columnlabels
modoutfile instructionfile category
tc1-fwd._os tc1-fwd._os.ins obs
tc1-fwd.lst tc1-fwd.lst.ins obs
END MODEL_OUTPUT_FILES
```

3.5.3 Instruction input files

A standard file can have lines at the top that are to be skipped (the number of lines can be zero). Subsequently, the data need to be in columns separated by spaces or commas. Simulated values are read from each line of data, and always from the same column. Instruction files can not contain any comment lines or blank lines, even at the end of the file.

Jif @, a delimiter

StandardFile, a keyword for standard files that is basically files where all read data are in one block

Nskip is the number of lines to skip at the beginning of the file and can equal 0

ReadColumn is the column from which Ucode reads

Nread is the number of values, and therefore lines, to be read.

Instruction files for non-standard files:

3.6 *Parallelisation*

3.6.1 Nähkästchen

The parent folder contains

- a script called start_runners.sh, which calls the individual runners using a hard-coded path. If only Runner 1 is lost, you've usually forgotten to set the path for Runner 2 and beyond. The runners might then run in a different scenario → bad!
- another .sh file (e.g. _ucode_mcmc_lucia.sh); this file calls the .in file (e.g. lucia_mcmc_ucode.in), which controls the ucode run; the parallelization block must be entered correctly at the end (not path-specific).
- The runner folder

Each runner-folder most contain:

- runner software (1.6MB)
- jrunner.rdy, of 0B size
- lucia.sh; this file initiates *lucia.py* and upon end of program *convert_outputs.py*
- lucia.py (with hard-coded path to the runner folder!) and all necessary maps and weather data as well as iitial_IC_part.py, tpl, obs files etc.
- convert_output.py with relevant path. Extracts values from ‘ .out time series outputs’
- the actual scenario folder, e.g. Val_repeat2, which contains settings.xml and the maps folder.

3.6.2 Official

Both, model runs and Ucode can be parallelised.

Jrunner software needed for parallelisation. Dispatcher / Master is the CPU that coordinates runs, runners / slaves are the CPUs that carry out the actual model runs. Sequence diagram of communication:

Execution time can be improved by also using processor 1 for model runs. This works best when, on processor 1, the priority of UCODE_2005 is set to something less than the default priority. On windows operating systems, this can be accomplished as follows. Start UCODE_2005. Press cntl-alt-delete to get a menu and then click on “Task Manager.” Click the tab “Processes”. Click on the column header “Image Name” twice so that UCODE_2005 is listed toward the top. Along the top of the window, click on “View” to get a drop-down menu, and click “Select Columns”. In the right column, make sure the box to the left of “Base Priority” has a check in it. Click “OK” to get back to the list of processes. Enlarge the window as needed to view the column labeled “Base Priority” Locate UCODE_2005 and right click on it. Click “Set Priority” and choose “BelowNormal”.

UCODE_2005 needs to be able to write and read files in each runner directory. For each runner directory and the dispatcher directory, copy in all the files and directories needed to run the process model(s). The runner directories also need the program JRUNNER. Start at least one JRUNNER before stating UCODE_2005. Additional instances of JRUNNER may be started after UCODE_2005 is started.

3.6.3 Parallel_Control Input Block (Optional)

Parallel = yes.

Wait = 0.001 (default), time delay, in seconds, used in file management.

VerboseRunner = 3, gives out most relevant system messages

AutoStopRunners default = yes, means that runners stop once Ucode stops.

OperatingSystem, options 'Windows', 'DOS', 'Unix', or 'Linux'.

TimeoutFactor Factor that multiplies RUNTIME to determine if a model run is overdue. Default=3.0. Example:

```
BEGIN PARALLEL_CONTROL
OPERATINGSYSTEM=Linux
Parallel=yes
Wait=.001
VerboseRunner=3
AutoStopRunners=yes
END PARALLEL_CONTROL
```

3.6.4 Parallel_Runners Input Block (Optional)

RunnerName (ID for runner / slaves), RunnerDir (each runner must be in a separate folder with all needed input files), RunTime (expected model runtime in seconds; if the model takes longer, the process will be aborted by the dispatcher.

RunnerName

RunnerDir: Pathname of directory where the runner program runs. The path needs to end with a backward or forward slash (\ or /), depending on the convention used by the operating system

RunTime: Expected model runtime, in seconds (to find problems with a runner, Time out factor = 3 x RunTime). Example:

```
BEGIN PARALLEL_RUNNERS TABLE
# RUNNERDIR must end with the correct directory separator for
# the OS -- "\" for Windows and "/" for Unix and Linux.
NROW=6 NCOL=4 COLUMNLABELS
RUNNERNAME RUNNERDIR RUNTIME
runner1 \dir\runner1\ 8000
runner2 \dir\runner2\ 8000
runner3 \dir\runner3\ 8000
runner4 \dir\runner4\ 8000
runner5 \dir\runner5\ 8000
runner6 \dir\runner6\ 8000
END PARALLEL_RUNNERS
```