A package of calibration procedures linked to SWAT through a generic platform to perform calibration, validation, and uncertainty analysis

Creator	K.C. Abbaspour, (Eawag)
Creation date	31.03.2012
Date of last revisions	20.05.2012
Subject	A software package for calibration of SWAT
Status	Finalized
Type	Software package
Description	A package of different calibration methods linked to SWAT
Contributors	K.C. Abbaspour, Eawag
Rights	Public
Identifier	EnviroGRIDS_D4.5
Language	English
Relation	EnviroGRIDS_D4.2, D4.3, D4.6, D4.7,D4.8

SWAT-CUP

SWAT-CUP (SWAT Calibration Uncertainty Procedures) is a package of software designed for calibration, validation, sensitivity analysis, and uncertainty analysis of a SWAT model. Details of the software is explained in the following pages which are the manual of the software.
The software can be downloaded from :
http://www.eawag.ch/forschung/siam/software/swat/index

DISCLAIMER

This report documents SWAT-CUP, a computer program for calibration of SWAT models. SWAT-CUP4 is a public domain program, and as such may be used and copied freely. The program links SUFI2, PSO, GLUE, ParaSol, and MCMC procedures to SWAT. It enables sensitivity analysis, calibration, validation, and uncertainty analysis of SWAT models. SWAT-CUP4 has been tested for all procedures prior to release. However, no warranty is given that the program is completely error-free. If you encounter problems with the code, find errors, or have suggestions for improvement, please contact Karim C. Abbaspour (abbaspour@eawag.ch) or Raghvan Srinivasan (r-srinivasan@tamu.edu).

New Implementations 2011

version 4.3.2

New features in this version are:

- Parallel processing (licensed)

- Visualization of the watershed outlets using the Bing map

- Formulation of multi-objective objective functions through extraction of variables from .rch, .hru, and .sub files

- Extraction and visualization of the 95ppu for variables from .rch, .hru, and .sub files when there are no observations

- Temperature data in tmp1.tmp is also allowed to be calibrated as well as the rainfall data

- In SUFI2, a threshold for the objective function can now be given so as to separate the behavioral solutions from the non-behavioral ones. The 95ppu as well as all other statistics are also calculated for these behavioral solutions.

- Input file formats have been changed to make the files more self-explanatory.

New Implementations 2010

version 3.1.1

1- Parameters of all soil layers can now be calibrated (see pages 32-34)

2- Next to landuse, texture, subbasin, and hydrologic unit, slope can also be used to parameterize the SWAT model

3- Management parameters can all be calibrated including each rotation and operation

4- All crop parameters can be explicitly calibrated

5- Rainfall in the file pcp.pcp can be calibrated for input uncertainty

6- At the end of the file *.gw, 20 auxiliary parameters can be specified as R1, R2, ...,R20, which can be used by other programs linked to SWAT. This is useful for users that link their own routines to SWAT and want to calibrate the parameters of their program along with the SWAT parameters.

7- Swat_EditLog.txt file lists the actual value of all the parameters that have been changed

8- GLUE, ParaSol, and MCMC now use the same *_extract_rch.def file as SUFI2 and can all accept missing observation data.

Content Page

Food for thought in calibration and application of watershed models

Calibration and uncertainty analysis of distributed watershed models is beset with a few serious issues that deserve the attention and careful consideration of researchers. These are: 1) Parameterization of watershed models. 2) Definition of what is a "calibrated watershed model" and what are the limits of its use. 3) Conditionality of a calibrated watershed model. 4) Calibration of highly managed watersheds where natural processes play a secondary role, and 5) uncertainty and non-uniqueness problems. These issues are briefly discussed here.

1) Model Parameterization

Should a soil unit appearing in various locations in a watershed, under different landuses and/or climate zones, have the same or different parameters? Probably it should have different parameters. The same argument could be made with all other distributed parameters. How far should one go with this differentiation? On the one hand we could have thousands of parameters to calibrate, and on other we may not have enough spatial resolution in the model to see the difference between different regions. This balance is not easy to determine and the choice of parameterization will affect the calibration results. Detailed information on spatial parameters is indispensable for building a correct watershed model. A combination of measured data and spatial analysis techniques using pedotransfer functions, geostatistical analysis, and remote sensing data would be the way forward.

2) When is a watershed model calibrated?

If a watershed model is calibrated using discharge data at the watershed outlet, can the model be called calibrated for that watershed? If we add water quality to the data and recalibrate, the hydrologic parameters obtained based on discharge alone will change. Is the new model now calibrated for that watershed? What if we add discharge data from stations inside the watershed? Will the new model give correct loads from various landuses in the watershed? Perhaps not, unless we include the loads in the calibration process (see Abbaspour et al., 2007). Hence, an important question arises as to: “for what purpose can we use a calibrated watershed model?” For example: What are the requirements of a calibrated watershed model if we want to do landuse change analysis? Or, climate change analysis? Or, analysis of upstream/downstream relations in water allocation and distribution? Can any single calibrated watershed model address all these issues? Can we have several calibrated models for the same watershed where each model is applicable to a certain objective? Note that these models will most likely have different parameters representing different processes (see Abbaspour et al. 1999).

3) Conditionality of calibrated watershed models

Conditionality is an important issue with calibrated models. This is related to the previous question on the limitation on the use of a calibrated model. Calibrated parameters are conditioned on the choice of objective function, the type, and numbers of data points and the procedure used for calibration, among other factors. In a previous study (Abbaspour et al. 1999), we investigated the consequences of using different variables and combination of variables from among pressure head, water content, and cumulative outflow on the estimation of hydraulic parameters by inverse modeling. The inverse study combined a global optimization procedure with a numerical solution of the one-dimensional variably saturated Richards flow equation. We analyzed multi-step drainage experiments with controlled boundary conditions in large lysimeters. Estimated hydraulic parameters based on different objective functions were all different from each other; however, a significant test of simulation results based on these parameters revealed that most of the parameter sets produced similar simulation results. Notwithstanding the significance test, ranking of the performances of the fitted parameters revealed that they were highly conditional with respect to the variables used in the objective function and the type of objective function itself. Mathematically, we could express a calibrated model M as:

$$M = M(\theta | p, g, w, b, v, m, ...)$$

where θ is a vector of parameters, p is a calibration procedure, g is the objective function type , w is a vector of weights in the objective function, b is the boundary conditions, *v* is the variables used in the objective function, *m* is the number of observed *v*'s, etc. Therefore, a calibrated model is conditioned on the procedure used for calibration, on the objective function, on the weights used in the objective function, on the initial and boundary conditions, on the type and length of measured data used in the calibration, etc. Such a model can clearly not be applied for just any scenario analysis.

4) Calibration of highly managed watersheds

In highly managed watersheds, natural processes play a secondary role. If detailed management data is not available, then modeling these watersheds will not be possible. Examples of managements are dams and reservoirs, water transfers, and irrigation from deep wells. In Figure 1a the effect of Aswan dam on downstream discharge before and after its operation is shown. It is clear that without the knowledge of dam's operation, it would not be possible to model downstream processes. Figure 1b shows the effect of wetland on discharge upstream, in the middle, and downstream of Niger Inland Delta.

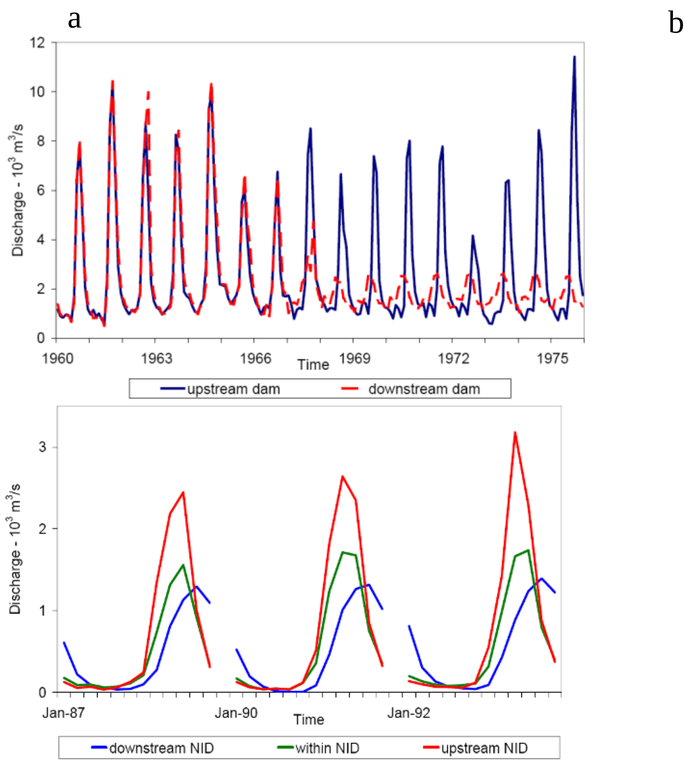


Figure 1. a) Effect of Aswan dam on down stream discharge before and after its operation in 1967. b) The influence of Niger Inland Delta on the through flow at upstream, within, and downstream of the wetland. (After Schuol et al., 2008a,b)

In Figure 2 the effect of irrigation on actual ET and soil moisture is illustrated in Esfahan, Iran. Esfahan is a region of high irrigation with a negative water balance for almost half of the year.

marf

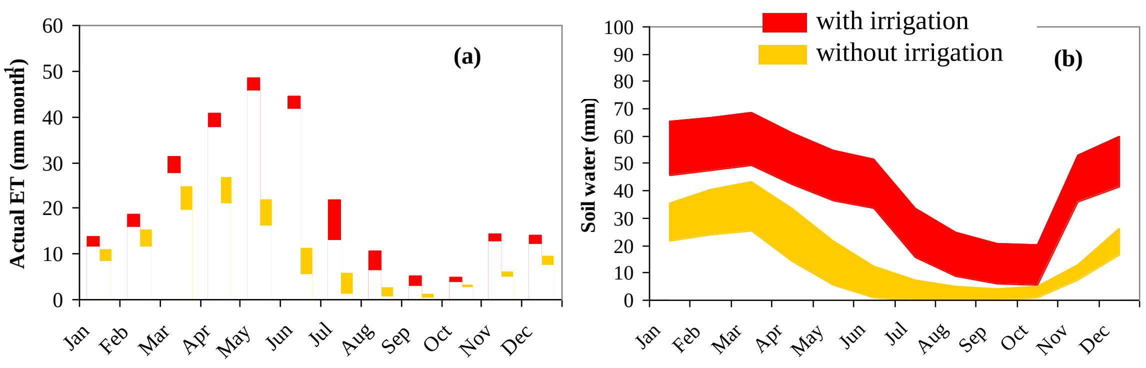


Figure 2. Illustration of the differences in predicted actual ET (a) and soil moisture (b) with and without considering irrigation in Esfahan province, Iran. The variables are monthly averages for the period of 1990-2002. (After Faramarzi et al., 2008)

In the study of water resources in Iran, Faramarzi et al., (2008) produced a "water management map" (Figure 3) in order to explain the calibration results of a hydrologic model of the country.

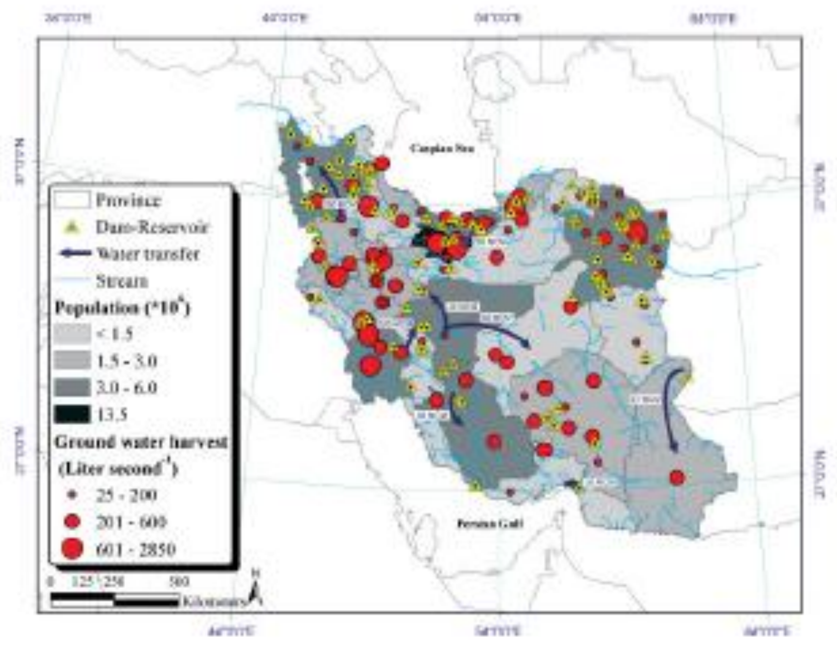


Figure 3. Water management map of Iran showing some of man's activities during 1990-2002. The map shows locations of dams, reservoir, water transfers and groundwater harvest (background shows provincial-based population). After Faramarzi et al., (2008).

5) Uncertainty issues

Another issue with calibration of watershed models is that of uncertainty in the predictions. Watershed models suffer from large model uncertainties. These can be divided into: conceptual model uncertainty, input uncertainty, and parameter uncertainty. The conceptual model uncertainty (or structural uncertainty) could be of the following situations: a) Model uncertainties due to simplifications in the conceptual model, b) Model uncertainties due to processes occurring in the watershed but not included in the model, c) Model uncertainties due to processes that are included in the model, but their occurrences in the watershed are unknown to the modeler, and d) Model uncertainties due to processes unknown to the modeler and not included in the model either!

Input uncertainty is as a result of errors in input data such as rainfall, and more importantly, extension of point data to large areas in distributed models. Parameter uncertainty is usually caused as a result of inherent non-uniqueness of parameters in inverse modeling. Parameters represent processes. The fact that processes can compensate for each other gives rise to many sets of parameters that produce the same output signal. A short explanation of uncertainty issues is offered below.

5.1) Conceptual model uncertainty

a) Model uncertainties due to simplifications in the conceptual model. For example, the assumptions in the universal soil loss equation for estimating sediment loss, or the assumptions in calculating flow velocity in a river. Figures 4a and 4b show some graphical illustrations.

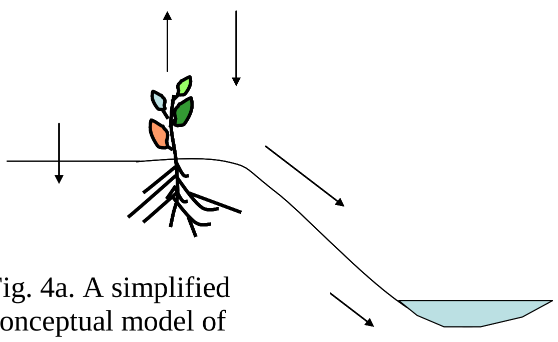


Fig. 4a. A simplified conceptual model of hydrology in a watershed where revap is ignored

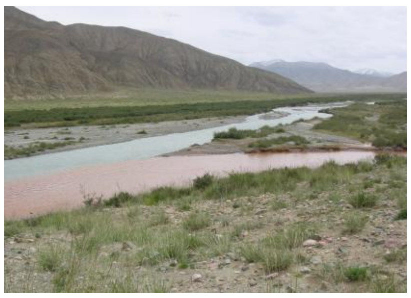


Fig. 4b. A natural process near the source of Yellow River in China playing havoc with river loading based on the USLE!

b) Model uncertainties due to processes occurring in the watershed but not included in the model. For example, wind erosion (Fig. 5a), erosions caused by landslides (Fig. 5b), and the "second-storm effect" effecting the mobilization of particulates from soil surface (see Abbaspour et al., 2007).

a

b

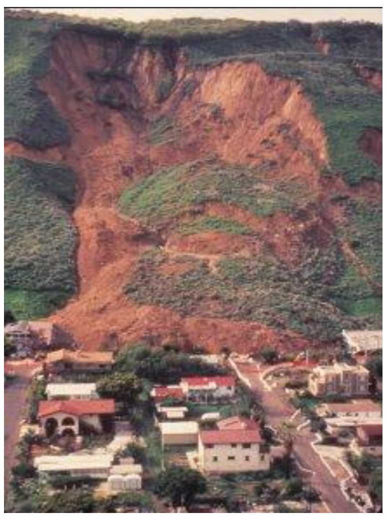


Figure 5. Natural processes not included in most watershed models but with a large impact on hydrology and water quality of a watershed, albeit for a short period

c) Model uncertainties due to processes that are included in the model, but their occurrences in the watershed are unknown to the modeler or unaccountable; for example, various forms of reservoirs, water transfer, irrigation, or farm management affecting water quality, etc. (Fig. 6, 7).

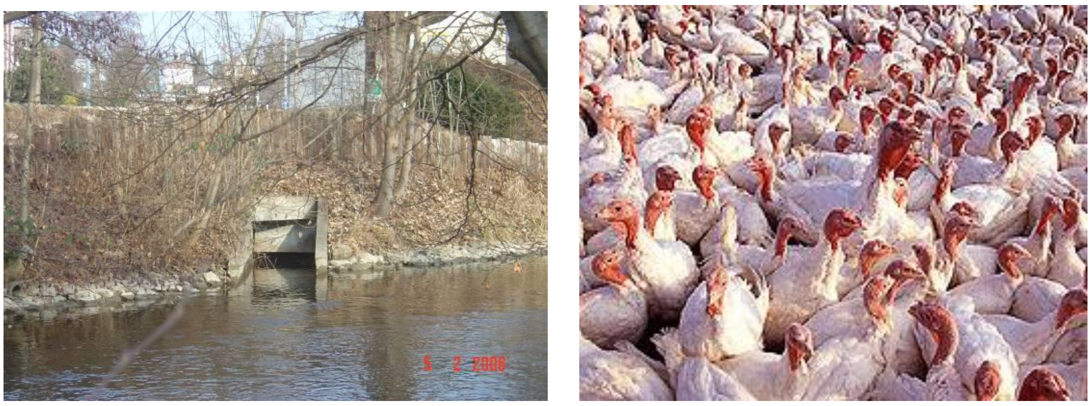


Fig. 6. Agricultural management practices such as water withdrawal and animal husbandry can affect water quantity and quality. These, may not always be known to the modeller.

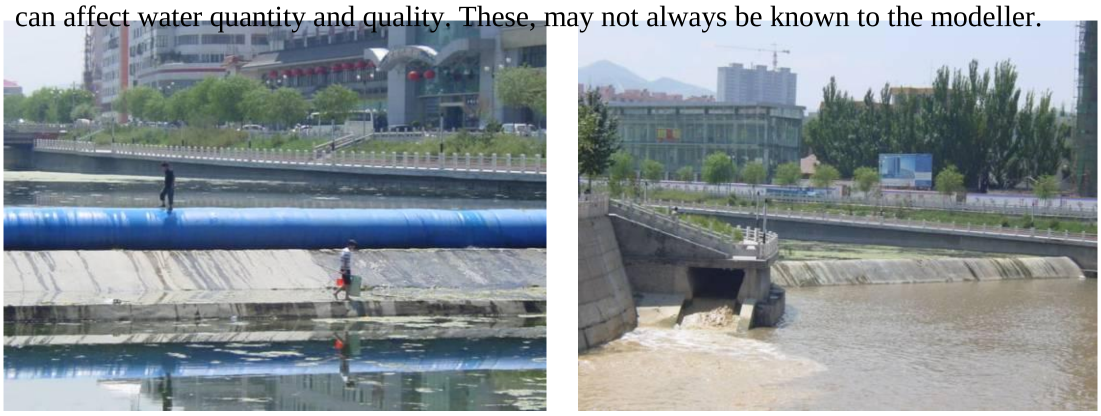


Fig. 7. Water control and water diversions may change the flow in ways that are unknown to the modeller and, hence, can not be accounted for in the model.

d) Model uncertainties due to processes unknown to the modeler and not included in the model either! These include dumping of waste material and chemicals in the rivers, or processes that may last for a number of years and drastically change the hydrology or water quality such as large-scale constructions of roads, dams, bridges, tunnels, etc. Figure 8 shows some situations that could add substantial “conceptual model error” to our analysis.

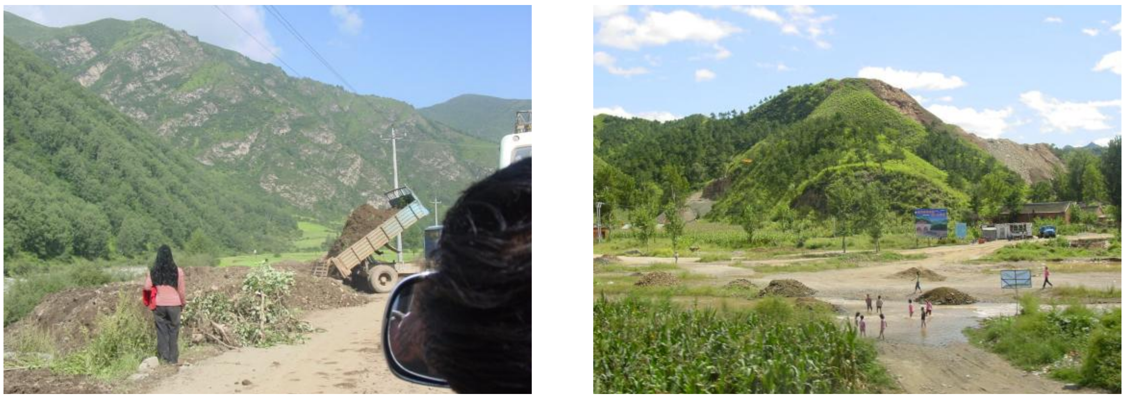


Fig. 8 Large construction projects such as roads, dams, tunnels, bridges, etc. can change river flow and water quality for a number of years. This may not be known or accountable by the modeller or the model

Fig. 8 continued

5.2) Input Uncertainty

In addition to model uncertainty, there are uncertainties due to errors in input variables such as rainfall and temperature, as point measurements are used in distributed models. It is quite difficult to account for input uncertainty. Some researchers propose treating inputs as random variable, which allows fitting them to get better simulations. As model outputs are very sensitive to input data, especially rainfall, care must be taken in such approaches. In mountainous regions, input uncertainty could be very large.

5.3) Parameter non-uniqueness

A single valued parameter results in a single model signal in direct modeling. In an inverse application, an observed signal, however, could be more-less reproduced with thousands of different parameter sets. This non-uniqueness is an inherent property of inverse modeling (IM). IM, has in recent years become a very popular method for calibration (e.g., Beven and Binley, 1992, 2001; Abbaspour et al., 1997, 2007; Duan et al., 2003; Gupta et al., 1998). IM is concerned with the problem of making inferences about physical systems from measured output variables of the model (e.g., river discharge, sediment concentration). This is attractive because direct measurement of parameters describing the physical system is time consuming, costly, tedious, and often has limited applicability. Because nearly all measurements are subject to some uncertainty, the inferences are usually statistical in nature.

GW_DELAY	CH_N2	CN2	REVAPMN	Sol_AWC	*g*
3.46	0.0098	50	0.8	0.11	0.010
0.34	0.131	20	2.4	0.2 3	0.011

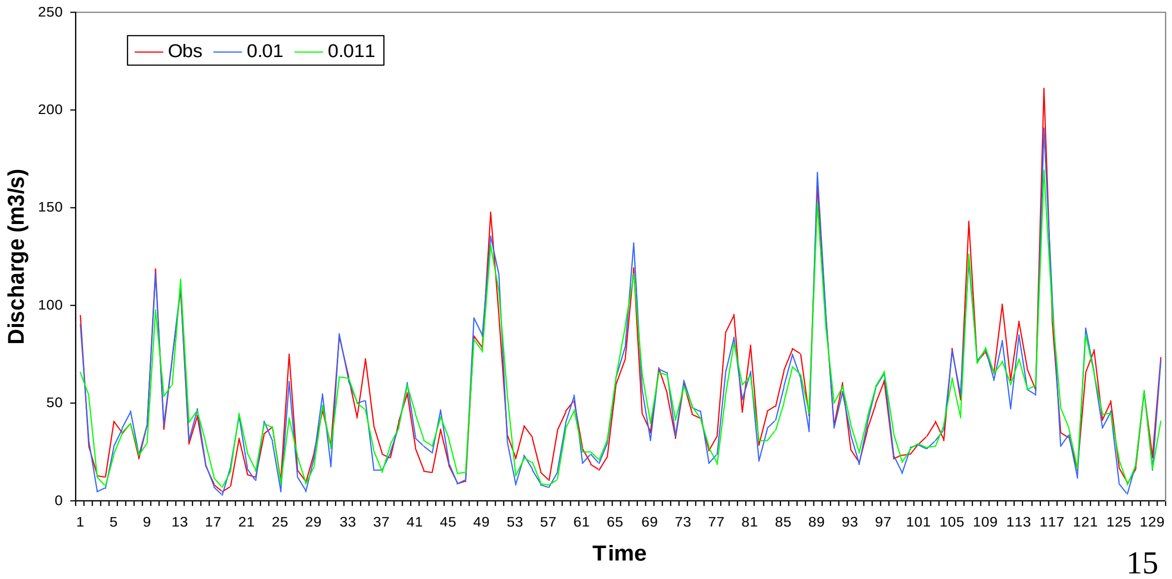


Figure 9. Example of parameter non-uniqueness showing two similar discharge signals based on quite different parameter values

Furthermore, because one can only measure a limited number of (noisy) data and because physical systems are usually modelled by continuum equations, no hydrological inverse problem is really uniquely solvable. In other words, if there is a single model that fits the measurements there will be many of them. An example is shown in Figure 9 where two very different parameter sets produce signals similar to the observed discharge. Our goal in inverse modelling is then to characterize the set of models, mainly through assigning distributions (uncertainties) to the parameters, which fit the data and satisfy our presumptions as well as other prior information.

The Swiss cheese effect

The non-uniqueness problem can also be looked at from the point of view of objective function. Plotting the objective-function response surface for two by two combinations of parameters could be quite revealing. As an example, see Figure 10 where the inverse of an objective function is plotted against two parameters, hence, local minima are shown as peaks. Size and distribution of these peaks resembles the mysterious holes in a block of Swiss Emmentaler cheese where the size of each hole represents the local uncertainty. Our experience shows that each calibration method converges to one such peak (see the papers by Yang et al., 2008, Schuol et al., 2008a, and Faramarzi et al., 2008). Yang et al., (2008) compared Generalized Likelihood Uncertainty Estimation (GLUE) (Beven and Binley, 1992), Parameter Solution (ParaSol) (Van Griensven and Meixner, 2003a), Sequential Uncertainty Fitting (SUFI2) (Abbaspour et al., 2004; 2007), and Markov chain Monte Carlo (MCMC) (e.g., Kuczera and Parent, 1998; Marshall et al., 2004; Vrugt et al., 2003; Yang et al., 2007) methods in an application to a watershed in China. They found that these different optimization programs each found a different solution at different locations in the parameter spaces with more less the same discharge results. Table 1 has a summary of the comparison.

To limit the non-uniqueness, the objective function should be made as comprehensive as possible by including different fluxes and loads (see Abbaspour et al., 2007). The downside of this is that a lot of data should be measured for calibration. The use of remote sensing data, when it becomes available, could be extremely useful. In fact we believe that the next big jump in watershed modeling will be made as a result of advances in remote sensing data availability.

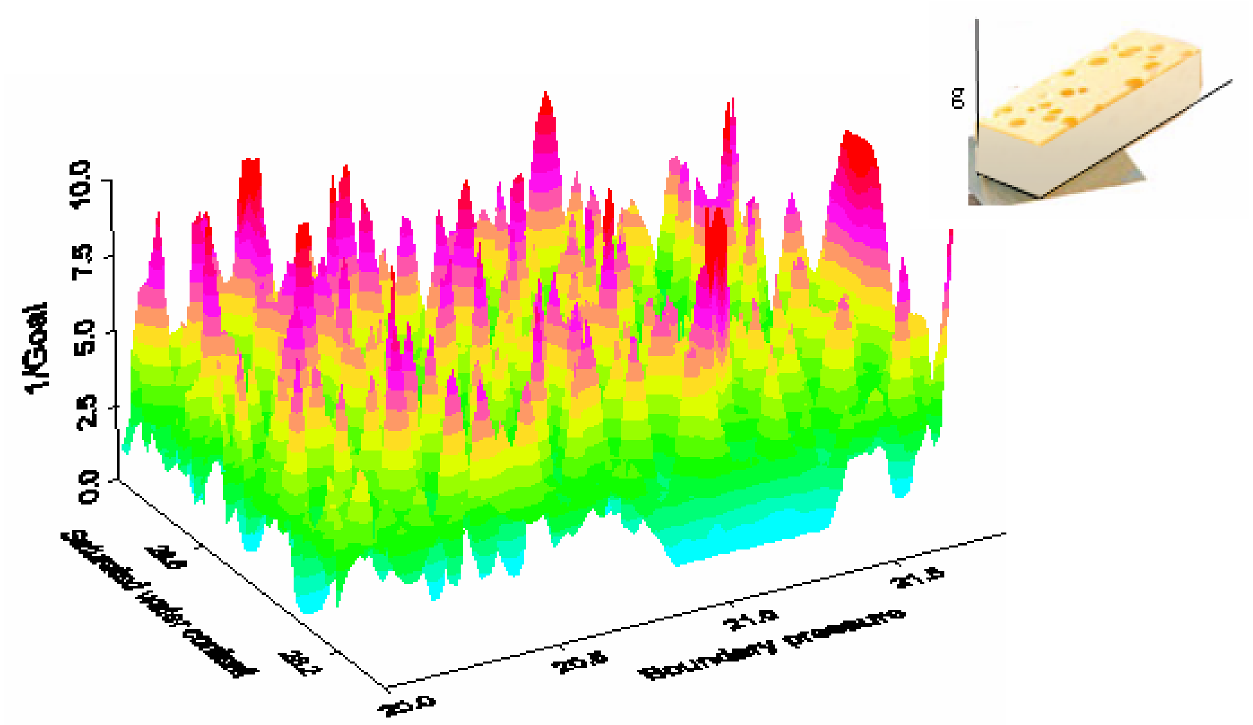


Figure 10. A multidimensional objective function is "multimodal" meaning that there are many areas of good solutions with different uncertainties much like the mysterious holes in a slice of Swiss cheese.

Further errors could also exist in the very measurements we use to calibrate the model. These errors could be very large, for example, in sediment data and grab samples if used for calibration. Another uncertainty worth mentioning is that of "modeler uncertainty"! It has been shown before that the experience of modelers could make a big difference in model calibration. We hope that packages like SWAT-CUP can help decrease modeler uncertainty by removing some probable sources of modeling and calibration errors.

On a final note, it is highly desirable to separate quantitatively the effect of different uncertainties on model outputs, but this is very difficult to do. The combined effect, however, should always be quantified on model out puts.

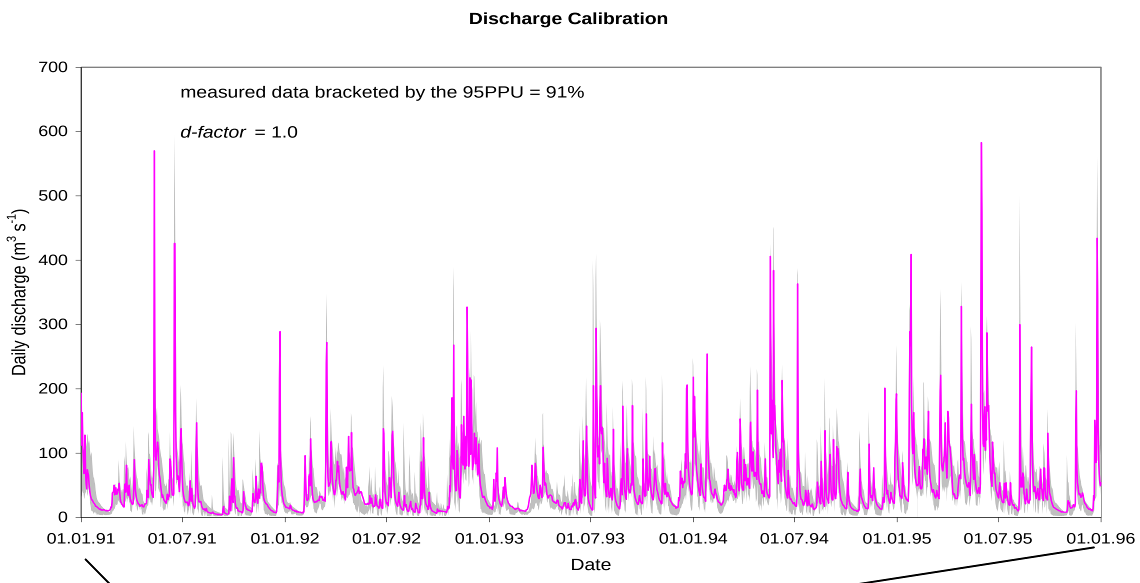
Table 1. Summary statistics comparing different calibration uncertainty procedures.

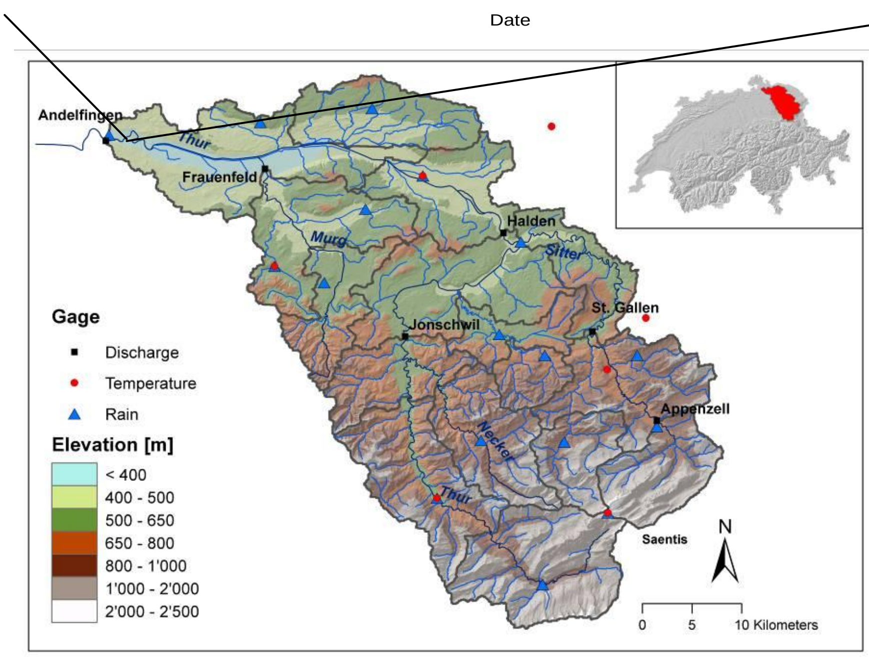
Criterion	GLUE	ParaSol	SUFI-2	Bayesian inference with cont. autoregr. error model	
				MCMC	IS
Goal function	Nash-Sutcliffe	Nash-Sutcliffe	Nash-Sutcliffe	post. prob. density	post. prob. density
a__CN2.mgt	-16.8 (-29.6, -9.8)[1]	-21.0 (-21.9, -20.1)	-26.9 (-30.0, -7.2)	-14.2 (-16.8, -11.6)	-19.60
v__ESCO.hru	0.76 (0.02, 0.97)	0.67 (0.65, 0.69)	0.82 (0.43, 1.0)	0.74 (0.63, 0.75)	0.62
v__EPCO.hru	0.22 (0.04, 0.90)	0.16 (0.13, 0.20)	1 (0.34, 1.0)	0.94 (0.39, 0.98)	0.27
r__SOL_K.sol	-0.16 (-0.36, 0.78)	-0.37 (-0.41, -0.34)	-0.1 (-0.58, 0.34)	-0.29 (-0.31, 0.78)	0.01
a__SOL_AWC.sol	0.11 (0.01, 0.15)	0.07 (0.08, 0.08)	0.07 (0.05, 0.15)	0.12 (0.1, 0.13)	0.05
v__ALPHA_BF.gw	0.12 (0.06, 0.97)	0.12 (0.08, 0.13)	0.51 (0.23, 0.74)	0.14 (0.11, 0.15)	0.91
v__GW_DELAY.gw	159.58 (9.7, 289.3)	107.7 (91.2,115.2)	190.07 (100.2, 300)	25.5 (17.8, 33.3)	33.15
r__SLSUBBSN.hru	-0.45 (-0.56, 0.46)	-0.59 (-0.60, -0.58)	-0.52 (-0.60, 0.03)	-0.55 (-0.56, 0.15)	0.58
a__CH_K2.rte	78.19 (6.0, 144.8)	35.70 (27.72,37.67)	83.95 (69.4, 150.0)	78.3 (68.0, 86.2)	147.23
a__OV_N.hru	0.05 (0.00, 0.20)	0.11 (0.07, 0.10)	0.06 (0.00, 0.11)	0.12 (0.00, 0.19)	0.08
[2]σ_{dry}	-	-	-	0.93 (0.81, 1.10)	0.87
[2]σ_{wet}	-	-	-	2.81 (2.4, 3.9)	2.30
[2]τ_{dry}	-	-	-	38.13 (29.5, 53.8)	28.47
[2]τ_{wet}	-	-	-	3.42 (2.4, 8.0)	0.92
NS for cal (val)	0.80 (0.78)	0.82 (0.81)	0.80 (0.75)	0.77 (0.77)	0.64 (0.71)
R^2 for cal (val)	0.80 (0.84)	0.82 (0.85)	0.81 (0.81)	0.78 (0.81)	0.70 (0.72)
LogPDF for cal (val)	-1989 (-926)	-2049 (-1043)	-2426 (-1095)	-1521 (-866)	-1650 (-801)
[3]*p-factor* for cal (val)	79% (69%)	18% (20%)	84% (82%)	85% (84%)	-
[4]*d-factor* for cal (val)	0.65 (0.51)	0.08 (0.07)	1.03 (0.82)	1.47 (1.19)	-
Uncertainty described by parameter uncertainty	All sources of uncertainty	Parameter uncertainty only	All sources of uncertainty	Parameter uncertainty only	Parameter uncertainty only
Difficulty of implement.	very easy	easy	easy	more complicated	more complicated
Number of runs	10000	7500	1500 + 1500	5000 + 20'000 + 20'000	100'000

[1] c(a, b) for each parameter means: c is the best parameter estimate, (a,b) is the 95% parameter uncertainty range except SUFI-2 (in SUFI-2, this interval denotes the final parameter distribution).
[2] the σ_{dry}, σ_{wet}, τ_{dry}, and τ_{wet} used to calculate the Calculate the logarithm of the posterior probability density function (PDF) are from the best of MCMC.
[3] *p-factor* means the percentage of observations covered by the 95PPU
[4] *d-factor* means relative width of 95% probability band
(After Yang et al., 2008).

SUFI2
Sequential Uncertainty Fitting

version 2





Conceptual basis of the SUFI-2 uncertainty analysis routine

In SUFI-2, parameter uncertainty accounts for all sources of uncertainties such as uncertainty in driving variables (e.g., rainfall), conceptual model, parameters, and measured data. The degree to which all uncertainties are accounted for is quantified by a measure referred to as the *P-factor*, which is the percentage of measured data bracketed by the 95% prediction uncertainty (95PPU). As all the processes and model inputs such as rainfall and temperature distributions are correctly manifested in the model output (which is measured with some error) - the degree to which we cannot account for the measurements - the model is in error; hence uncertain in its prediction. Therefore, the percentage of data captured (bracketed) by the prediction uncertainty is a good measure to assess the strength of our uncertainty analysis. The 95PPU is calculated at the 2.5% and 97.5% levels of the cumulative distribution of an output variable obtained through Latin hypercube sampling, disallowing 5% of the very bad simulations. As all forms of uncertainties are reflected in the measured variables (e.g., discharge), the parameter uncertainties generating the 95PPU account for all uncertainties. Breaking down the total uncertainty into its various components is highly interesting, but quite difficult to do, and as far as the author is aware, no reliable procedure yet exists.

Another measure quantifying the strength of a calibration/uncertainty analysis is the *R-factor*, which is the average thickness of the 95PPU band divided by the standard deviation of the measured data. SUFI-2, hence seeks to bracket most of the measured data with the smallest possible uncertainty band. The concept behind the uncertainty analysis of the SUFI-2 algorithm is depicted graphically in Figure 11. This Figure illustrates that a single parameter value (shown by a point) leads to a single model response (Fig. 11a), while propagation of the uncertainty in a parameter (shown by a line) leads to the 95PPU illustrated by the shaded region in Figure 11b. As parameter uncertainty increases, the output uncertainty also increases (not necessarily linearly) (Fig. 11c). Hence, SUFI-2 starts by assuming a large parameter uncertainty (within a physically meaningful range), so that the measured data initially falls within the 95PPU, then decreases this uncertainty in steps while monitoring the *P-factor* and the *R-factor*. In each step, previous parameter ranges are updated by calculating the sensitivity matrix (equivalent to Jacobian), and equivalent of a Hessian matrix, followed by the calculation of covariance matrix, 95% confidence intervals of the parameters, and correlation matrix. Parameters are then updated in such a way that the new ranges are always smaller than the previous ranges, and are centered around the best simulation (for more detail see Abbaspour et al., 2004, 2007).

The goodness of fit and the degree to which the calibrated model accounts for the uncertainties are assessed by the above two measures. Theoretically, the value for *P-factor* ranges between 0 and 100%, while that of *R-factor* ranges between 0 and infinity. A *P-factor* of 1 and *R-factor* of zero is a simulation that exactly corresponds to measured data. The degree to which we are away from these numbers can be used to judge the strength of our calibration. A larger *P-factor* can be achieved at the expense of a larger *R-factor*.

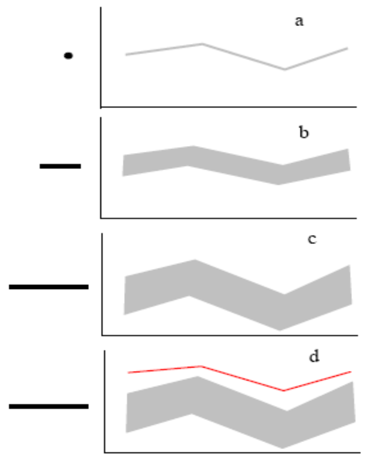


Figure 11. A conceptual illustration of the relationship between parameter uncertainty and prediction uncertainty

Hence, often a balance must be reached between the two. When acceptable values of *R-factor* and *P-factor* are reached, then the parameter uncertainties are the desired parameter ranges. Further goodness of fit can be quantified by the R^2 and/or Nash-Sutcliff (*NS*) coefficient between the observations and the final "best" simulation. It should be noted that we do not seek the "best simulation" as in such a stochastic procedure the "best solution" is actually the final parameter ranges.

If initially we set parameter ranges equal to the maximum physically meaningful ranges and still cannot find a 95PPU that brackets any or most of the data, for example, if the situation in Figure 11d occurs, then the problem is not one of parameter calibration and the conceptual model must be re-examined.

SUFI-2 as an optimization algorithm

A short step-by-step description of SUFI-2 algorithm is as follows:

Step 1. In the first step an objective function is defined. The literature shows many different ways of formulating an objective function (e.g., Legates and McCabe, 1999; Gupta et al., 1998). Each formulation may lead to a different result; hence, the final parameter ranges are always conditioned on the form of the objective function. To overcome this problem, some studies (e.g., Yapo et al., 1998) combine different types of functions (e.g., based on root mean square error, absolute difference, logarithm of differences, R^2, Chi square, Nash-Sutcliffe, etc.) to yield a "multi-criteria" formulation. The use of a "multi-objective" formulation (Duan et al. 2003; Gupta et al., 1998) where different variables are included in the objective function is also important to reduce the non-uniqueness problem. The objective functions included in SUFI-2 are described later in the manual.

Step 2. The second step establishes physically meaningful absolute minimum and maximum ranges for the parameters being optimized. There is no theoretical basis for excluding any one particular distribution. However, because of the lack of information, we assume that all parameters are uniformly distributed within a region bounded by minimum and maximum values. Because the absolute parameter ranges play a constraining role, they should be as large as possible, yet physically meaningful:

$$b_j: b_{j,\text{abs\_min}} \le b_j \le b_{j,\text{abs\_max}} \qquad j = 1.... m, \qquad (1)$$

where b_j is the j-th parameter and m is the number of parameters to be estimated.

Step 3. This step involves an optional, yet highly recommended "absolute sensitivity analysis" for all parameters in the early stages of calibration. We maintain that no automated optimization routine can replace the insight from physical understanding and knowledge of the effects of parameters on the system response. The sensitivity analysis is carried out by keeping all parameters constant to realistic values, while varying each parameter within the range assigned in step one. Plotting results of these simulations along with the observations on the same graph gives insight into the effects of parameter changes on observed signals.

Step 4. Initial uncertainty ranges are next assigned to parameters for the first round of Latin hypercube sampling, i.e,

$$b_j: [b_{j,\min} \leq b_j \leq b_{j,\max}] \qquad j = 1, m \tag{2}$$

In general, the above ranges are smaller than the absolute ranges, are subjective, and depend upon experience. The sensitivity analysis in step 3 can provide a valuable guide for selecting appropriate ranges. Although important, these initial estimates are not crucial as they are updated and allowed to change within the absolute ranges.

Step 5. Latin Hypercube (McKay et al., 1979) sampling is carried out next; leading to *n* parameter combinations, where *n* is the number of desired simulations. This number should be relatively large (approximately 500-1000). The simulation program is then run *n* times and the simulated output variable(s) of interest, corresponding to the measurements, are saved.

Step 6. As a first step in assessing the simulations, the objective function, *g*, is calculated.

Step 7: In this step a series of measures is calculated to evaluate each sampling round. First, the sensitivity matrix, **J**, of $g(\mathbf{b})$ is computed using:

$$J_{\mathrm{ij}} = \frac{\Delta g_i}{\Delta b_j} \qquad i = 1,\ldots,\ C_2^n, \quad j = 1,\ldots,\ m, \tag{3}$$

where C_2^n is the number of rows in the sensitivity matrix (equal to all possible combinations of two simulations), and *j* is the number of columns (number of parameters). Next, equivalent of a Hessian matrix, **H**, is calculated by following the Gauss-Newton method and neglecting the higher-order derivatives as:

$$\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J} \tag{4}$$

Based on the Cramer-Rao theorem (Press et al., 1992) an estimate of the lower bound of the parameter covariance matrix, **C**, is calculated from:

$$\mathbf{C} = s_g^2 \left(\mathbf{J}^T \mathbf{J}\right)^{-1}, \tag{5}$$

where s_g^2 is the variance of the objective function values resulting from the *n* runs. The estimated standard deviation and 95% confidence interval of a parameter b_j is calculated from the diagonal elements of **C** (Press et al., 1992) from:

$$s_j = \sqrt{C_{jj}} \tag{6}$$

$$b_{j,\text{lower}} = b_j^* - t_{\nu,0.025} s_j \tag{7}$$

$$b_{j,\text{upper}} = b_j^* + t_{\nu,0.025} s_j , \tag{8}$$

where b_j^* is the parameter *b* for one of the best solutions (i.e., parameters which produce the smallest value of the objective function), and ν is the degrees of freedom (*n* – *m*). Parameter correlations can then be assessed using the diagonal and off-diagonal terms of the covariance matrix as follows:

$$r_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}}\sqrt{C_{jj}}} \tag{9}$$

It is important to note that the correlation matrix **r** quantifies the change in the objective function as a result of a change in parameter *i*, relative to changes in the other parameters *j*. As all parameters are allowed to change, the correlation between any two parameters is quite small. This is expected because in SUFI-2 sets of parameters are drawn contrary to procedures that keep all parameters constant while changes only one.

Parameter sensitivities were calculated by calculating the following multiple regression system, which regresses the Latin hypercube generated parameters against the objective function values:

$$g = \alpha + \sum_{i=1}^{m} \beta_i b_i \tag{10}$$

A *t*-test is then used to identify the relative significance of each parameter b_i. We emphasize that the measures of sensitivity given by [10] are different from the sensitivities calculated in step 3. The sensitivities given by [10] are estimates of the average changes in the objective function resulting from changes in each parameter, while all other parameters are changing. Therefore, [10] gives relative sensitivities based on linear approximations and, hence, only provides partial information about the sensitivity of the objective function to model parameters. Furthermore, the relative sensitivities of different parameters, as indicated by the t-test, depend on the ranges of the parameters. Therefore, the ranking of sensitive parameters may changes in every iteration.

Step 8. In this step measures assessing the uncertainties are calculated. Because SUFI-2 is a stochastic procedure, statistics such as percent error, R^2, and Nash-Sutcliffe, which compare two signals, are not applicable. Instead, we calculate the 95% prediction uncertainties (95PPU) for all the variable(s) in the objective function. As previously mentioned, this is calculated by the 2.5th (X_L) and 97.5th (X_U) percentiles of the cumulative distribution of every simulated point. The goodness of fit is, therefore, assessed by the uncertainty measures calculated from the percentage of measured data bracketed by the 95PPU band, and the average distance $\bar{d}$ between the upper and the lower 95PPU (or the degree of uncertainty) determined from:

$$\bar{d}_X = \frac{1}{k}\sum_{l=1}^{k}(X_U - X_L)_l \,, \qquad (11)$$

where k is the number of observed data points. The best outcome is that 100% of the measurements are bracketed by the 95PPU, and $\bar{d}$ is close to zero. However, because of measurement errors and model uncertainties, the ideal values will generally not be achieved. A reasonable measure for $\bar{d}$, based on our experience, is calculated by the *R-factor* expressed as:

$$R\text{-}factor = \frac{\bar{d}_X}{\sigma_X} \,, \qquad (12)$$

where σ_X is the standard deviation of the measured variable X. A value of less than 1 is a desirable measure for the *R-factor*.

Step 9: Because parameter uncertainties are initially large, the value of $\bar{d}$ tends to be quite large during the first sampling round. Hence, further sampling rounds are needed with updated parameter ranges calculated from:

$$b'_{j,\min} = b_{j,lower} - \mathrm{Max}\left(\frac{(b_{j,\mathrm{lower}} - b_{j,\min})}{2}, \frac{(b_{j,\max} - b_{j,\mathrm{upper}})}{2} \right)$$

$$b'_{j,\max} = b_{j,\mathrm{upper}} + \mathrm{Max}\left(\frac{(b_{j,\mathrm{lower}} - b_{j,\min})}{2}, \frac{(b_{j,\max} - b_{j,\mathrm{upper}})}{2} \right),$$

(13)

where b' indicate updated values. Parameters of the best simulation is used to calculate $b_{j,\mathrm{lower}}$ and $b_{j,\mathrm{upper}}$. The above criteria, while producing narrower parameter ranges for subsequent iterations, ensure that the updated parameter ranges are always centered on the best estimates. In the formulation of 13, the uncertainty in the sensitive parameters reduces faster than those of the insensitive parameters due to the inclusion of the confidence interval, which is larger for less sensitive parameters.

SWAT-CUP4

Automated model calibration requires that the uncertain model parameters are systematically changed, the model is run, and the required outputs (corresponding to measured data) are extracted from the model output files. The main function of an interface is to provide a link between the input/output of a calibration program and the model. The simplest way of handling the file exchange is through text file formats.

SWAT-CUP is an interface that was developed for SWAT. Using this generic interface, any calibration/uncertainty or sensitivity program can easily be linked to SWAT. A schematic of the linkage between SWAT and SUFI2 is illustrated in Figure 12.

A step by step operation of SWAT-SUFI2 is given below.

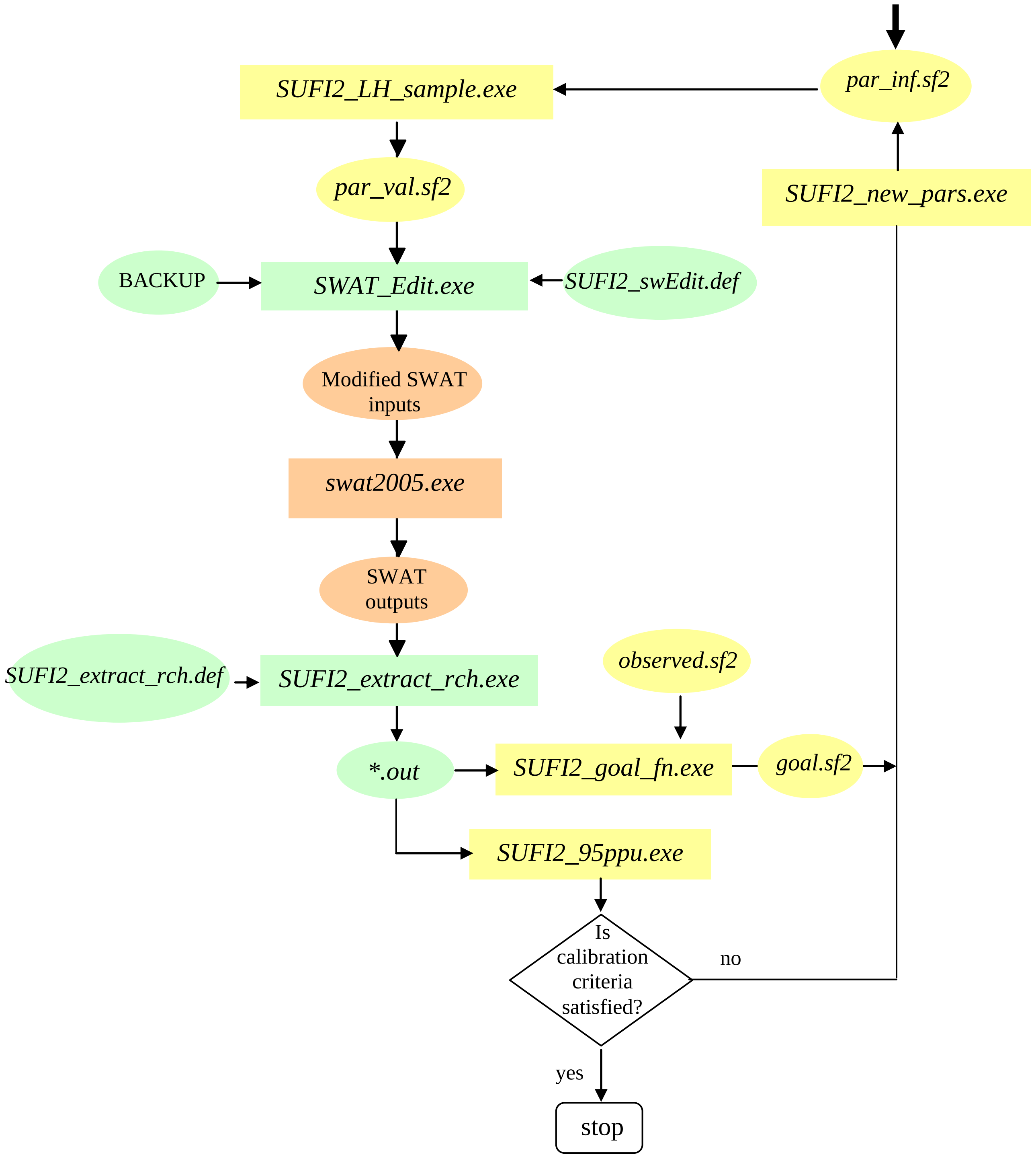


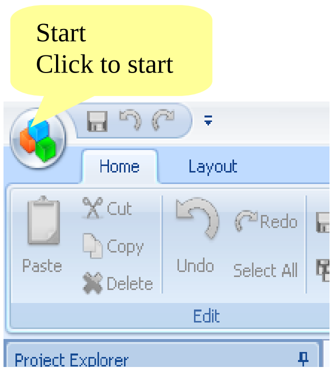
Figure 12. Showing the link between SWAT (orange), iSWAT (green), and SUFI2 (yellow)
The entire algorithm is run by two batch files: *SUFI2_pre.bat* and *SUFI2_post.bat*

Step-by-step running of SWAT-SUFI2

1- Read the theory and application of SWAT-SUFI2 at the beginning of this manual and in the following papers:

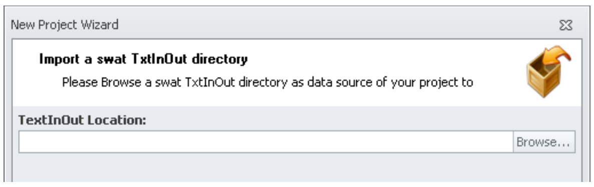
- Thur watershed paper (Abbaspour et al., 2007)
- The application to the landfills in Switzerland (Abbaspour et al., 2004)
- The continental application in Africa (Schuol et al, 2008a,b)
- The country-based application to Iran (Faramarzi et al., 2008)
- The comparison of different programs (Yang et al., 2008)

2. Install the SWAT-CUP and start the program



3. For a new project:

a) Locate a "TxtInOut" directory. Any file with "TxtInOut" in the name string would be accepted


New Project Wizard

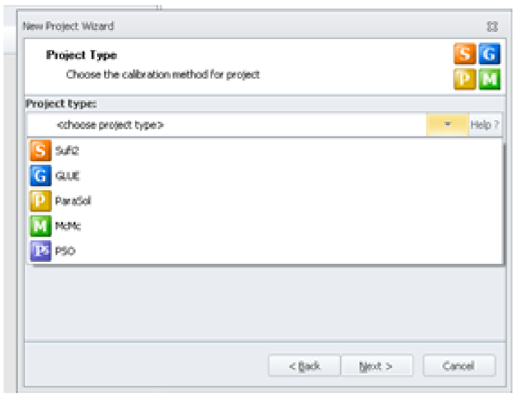
Import a swat TxtInOut directory

Please Browse a swat TxtInOut directory as data source of your project to

TextInOut Location:

Browse...

b) Select a program from the list provided (SUFI2, GLUE, ParaSol, MCMC, PSO).


New Project Wizard

Project Type

Choose the calibration method for project

Project type:

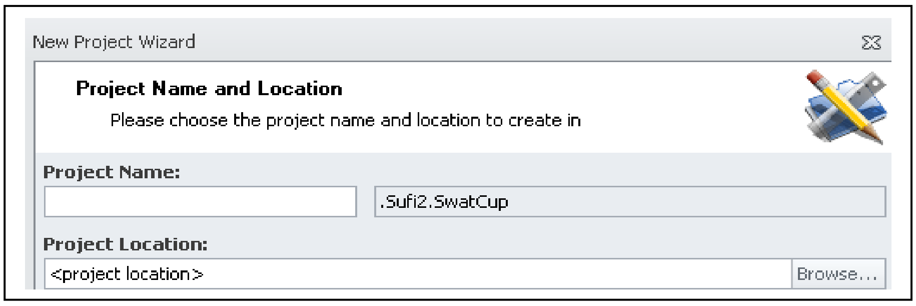
<choose project type>

Help ?

Sufi2
GLUE
ParaSol
McMc
PSO

< Back   Next >   Cancel

c) Give a name to the project. Note the default addition to the name provided in the window to the right of "Project Name" window.


New Project Wizard

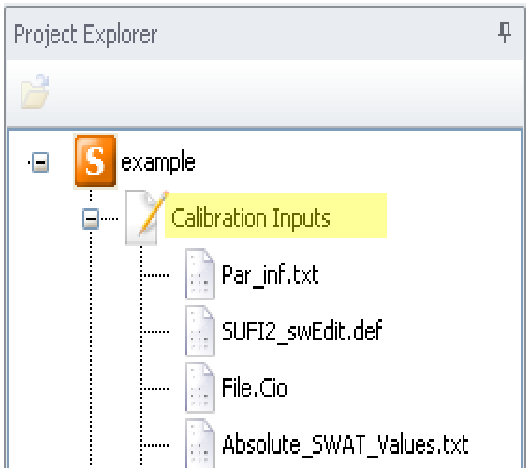
Project Name and Location
Please choose the project name and location to create in

Project Name: .Sufi2.SwatCup

Project Location: <project location>

The program creates the desired project directory and copies there all TxtInOut files. It also creates a directory called "Backup" and copies all SWAT file there. The parameters in the files in the Backup directory serve as the default parameters and do not changed.

4. Under the **Calibration Inputs** edit the following files:



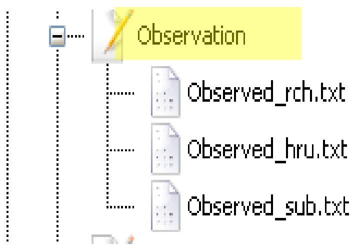
- ***Par_inf.txt*** file - contains input parameters to be optimized. This file needs to be edited by the user. Examples show the formats. Edit this to your needs.
Parameterization is explained below.

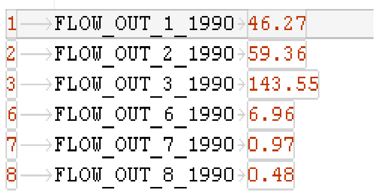
- ***SUFI2_swEdit.def*** - contains the beginning and the ending simulations.

- ***File.cio*** - This is a SWAT file. It is put here for convenience. What you need from this file are the simulation years and the number of warm-up period (NYSKIP) to correctly provide SWAT-CUP with beginning and end year of simulation (not including warm-up period).

- ***Absolute_SWAT_Values.txt*** - All parameters to be fitted should be in this file plus their absolute min and max ranges. Currently most, but not all parameters are included in this file. Simply add to it the parameters that don't exist in the format that is shown.

5. Under **Observation** are three files that contain the observed variables. Observed variables correspond to the variables in output.rch, output.hru, and output.sub files. Variables from different files can now be included to form a multi-objective objective function. Simply **only edit** the file(s) that applies to your project and do not worry about the ones that don't. The format should be exactly as shown in the examples that are included in the program. You may have missing data here that can be expressed as shown in the example files. The first column has sequential numbers from the beginning of simulation time period (in the example here days 4 and 5 are missing). Second column has an arbitrary format. Here, it is showing the

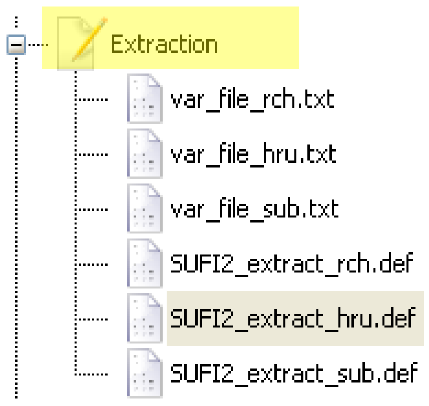




```
1	FLOW_OUT_1_1990	46.27
2	FLOW_OUT_2_1990	59.36
3	FLOW_OUT_3_1990	143.55
6	FLOW_OUT_6_1990	6.96
7	FLOW_OUT_7_1990	0.97
8	FLOW_OUT_8_1990	0.48
```

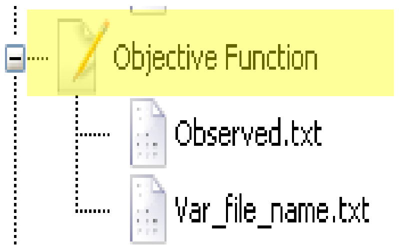
variable name, month, and year. Third column is the variable value.

6. Under **Extraction** you will find two types of files **.txt** and **.def** corresponding again to SWAT output files output.rch, output.hru, and output.sub. If you have observations corresponding to variables in these files, then you need to extract the corresponding simulated values from these files.



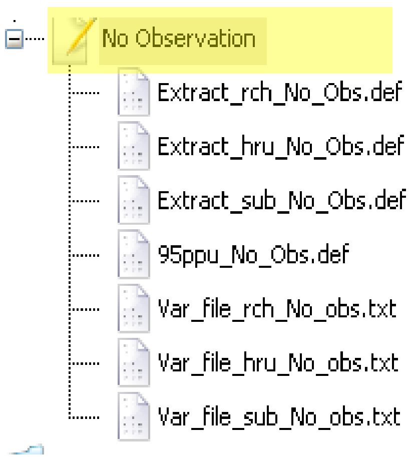
.txt files simply contain the names of the files that extracted values should be written to, and **.def** files define which variables need to be extracted from which subbasins. These files are relatively self-explanatory. Here again **only edit** the needed files.

7. Next, **Objective Function** is defined. In this step the file **observed.txt** should be edited. This file contains all the information in **observed_rch.txt**, **observed_hru.txt**, **observed_sub.txt** files, plus some extra information for the calculation of objective function.

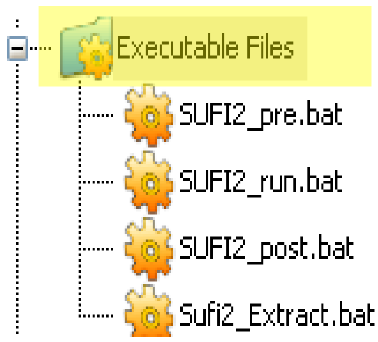


Similarly, the **Var_file_name.txt** contains the names of all the variables that should be included in the in the objective function. These names are similar to the names in the var_file_*.txt in the **Extraction** section.

8. The **No Observation** section is designed for the extraction and visualization of uncertainties in the variables for which we have no observations, but would like to see how they are simulated such as various nutrient loads, or soil moisture, etc. The **.txt** and **.def** files are the same as the **Extraction** section. The file **95ppu_No_Obs.def** is a file used for the calculation of the 95ppu in the extracted variables with no observation. All these files are self explanatory as they appear in the examples provided.


No Observation
Extract_rch_No_Obs.def
Extract_hru_No_Obs.def
Extract_sub_No_Obs.def
95ppu_No_Obs.def
Var_file_rch_No_obs.txt
Var_file_hru_No_obs.txt
Var_file_sub_No_obs.txt

9. The section **Executable Files** plays the role of engine in SWAT-CUP. The four batch files indicate what should or should not be run.


Executable Files
SUFI2_pre.bat
SUFI2_run.bat
SUFI2_post.bat
Sufi2_Extract.bat

SUFI2_pre.bat - this batch file runs the pre-processing procedures, which now include running the Latin hypercube sampling program. This batch file usually does not need to be edited.

SUFI2_pre.bat
```
SUFI2_LH_sample.exe
```

SUFI2_run.bat - this program executes **SUFI2_execute.exe** program, which runs the **SWAT_Edit.exe**, extraction batch files, as well as **SWAT2009.exe**.

SUFI2_run.bat
```
sufi2_execute.exe
```

SUFI2_post.bat - runs the post-processing batch file, which runs the programs for objective function calculation, new parameter calculation, 95ppu calculation, 95ppu for behavioral simulations, and 95ppu for the variables with no observations.

SUFI2_post.bat

```
SUFI2_goal_fn.exe
SUFI2_new_pars.exe
SUFI2_95ppu.exe
SUFI2_95ppu_beh.exe


95ppu_NO_Obs.exe
```

SUFI2_Extract.bat - This batch file should contain the names of all the extract programs with or without observations.

Currently 6 programs can be supported:

SUFI2_Extract.bat

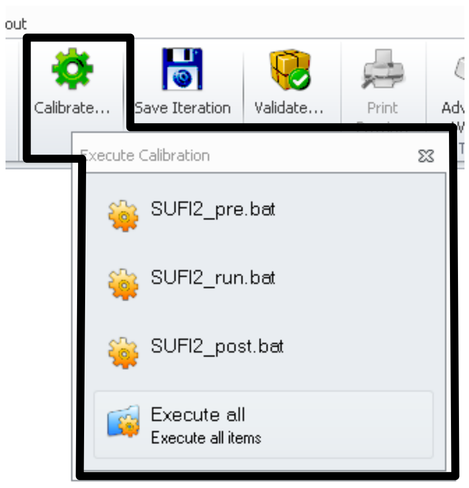
```
SUFI2_extract_rch.exe
SUFI2_extract_hru.exe
SUFI2_extract_sub.exe

extract_hru_No_Obs.exe
extract_rch_No_Obs.exe
extract_sub_No_Obs.exe
```

This file must be edited and the programs that are not desired to run should be “remarked” as shown below:

SUFI2_Extract.bat

```
SUFI2_extract_rch.exe

rem  SUFI2_extract_hru.exe
rem  SUFI2_extract_sub.exe

rem  extract_hru_No_Obs.exe
rem  extract_rch_No_Obs.exe
rem  extract_sub_No_Obs.exe
```

10. Next, after editing all the input files, the programs in the **Calibration** are executed. In this section three steps are performed:

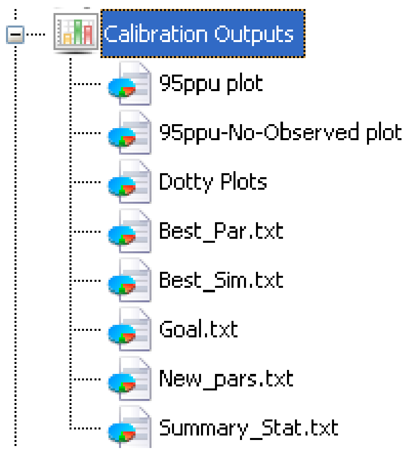


Sufi2_pre.bat - This command runs the **Sufi2_pre.bat** file. This file must be run before the start of every new iteration.

SUFI2_run.bat - This command executes the run batch file. This file is described above in 9.

SUFI2_post.bat - After all simulations are finished, this command executes the post processing batch file described above in 9.

11. **Calibration Outputs**



95ppu plot - This command shows the 95ppu of all variables. Also shown are observations and best simulation of the current iteration.

95ppu-No_Observed plot - This command shows the 95ppu of all variables with no observations

Dotty Plots - This command show the dotty plots of all parameters. These are plots of parameter values vs objective function. The main purpose of these graphs are to show the distribution of the sampling points as well as to give an idea of parameter sensitivity.

Best_Par.txt - This command shows the best parameters (giving the best value of objective function) of the current iteration.

Best_Sim.txt - This command shows the best simulated values.

Goal.txt - This command shows the value of all parameter sets as well as the value of the objective function in the last column.

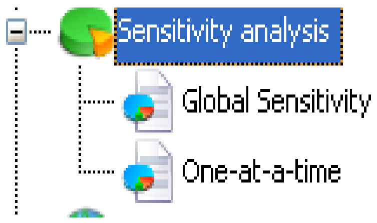
New_Pars.txt - This file shows the suggested values of the new parameters to be used in the next iteration. These values can be copied and pasted in the Par_inf.txt file for the next iteration. The new parameters should be checked for possible unreasonable values (e.g., negative hydraulic conductivity,..). These parameter values should be manually corrected.

Summary_Stat - This file has the statistics comparing observed data with the simulation band through p-factor and r-factor and the best simulation of the current iteration by using R2, NS, bR2, MSE, and SSQR. For definition of these functions see the section on objective functions. Also shown is the objective function (goal function) type, best simulation number of the current iteration, and the best value of the objective function for the current run.

If behavioral solutions exist, then the p-factor and r-factor for these solutions are also calculated. As shown in the following Table the effect of using behavioral solutions is to obtain smaller p-factor and r-factor, or a smaller prediction uncertainty.

```
Goal_type= br2 (type 6)   Best_sim_no= 12    Best_goal  = 2.793585e-
001

Variable     p-factor     r-factor   R2     NS     br2     MSE      SSQR
q_31            0.71        0.94      0.55   0.36   0.38   2004.1
534.2
q_43            0.70        0.96      0.33   0.00   0.18   4263.8
792.8
.....



---- Results for behavioral parameters ---
Behavioral threshold= 0.2
Number of behavioral simulations = 35

Variable     p_factor     r-factor    R2     NS     br2     MSE
SSQR
q_31            0.65         0.82      0.55   0.36   0.38   2004.1
534.2
q_43            0.62         0.78      0.33   0.00   0.18   4263.8
792.8
.....
```

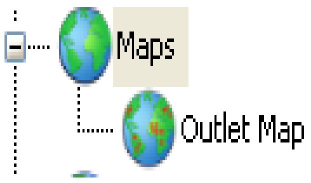
12. **Sensitivity analysis** - This section of the program performs sensitivity analysis. Two types of sensitivity analysis are allowed. **Global Sensitivity** and **One-at-a-time** sensitivity analysis.



Global sensitivity analysis can be performed after an iteration.

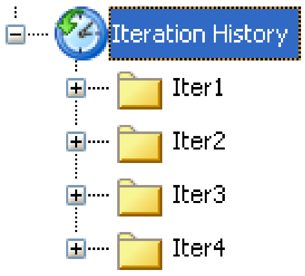
One-at-a-time sensitivity should be performed for one parameter at a time only. The procedure is explained in the next section.

13. **Maps** - Maps section refers to the visualization of the outlets. The Bing map is started and the location of outlets is projected on the map using their Lat, Long coordinate.



Both text and dbf files are accepted. The programs looks for a file called **outlets.txt** or **outlets.dbf**. The program converts these into **outlets.xml** file. If there is one such file in the swatcup project directory, then remove or rename it and locate the outlets file for the visualization.

14. **Iterations History** - All iterations can be saved in the iteration history and the progress to convergence studied.



15. After a complete iteration, review the suggested new parameters in the new_pars.txt, copy them to par_inf.txt and make a new iteration.

Parameterization in SWAT-CUP

In SWAT, the HRU is the smallest unit of spatial disaggregation. As a watershed is divided into HRUs based on elevation, soil, and landuse, a distributed parameter such as hydraulic conductivity can potentially be defined for each HRU. An analyst is, hence, confronted with the difficult task of collecting or estimating a large number of input parameters, which are usually not available. An alternative approach for the estimation of distributed parameters is by calibrating a single global modification term that can scale the initial estimates by a multiplicative, or an additive term. This leads to the proposed parameter identifiers.

An important consideration for applying parameter identifiers is that the changes made to the parameters should have physical meanings and should reflect physical factors such as soil, landuse, elevation, etc. Therefore, the following scheme is suggested:

x__<parname>.<ext>__<hydrogrp>__<soltext>__<landuse>__<subbsn>__<slope>

Where

x__ = Code to indicate the type of change to be applied to the parameter:

v__ means the existing parameter value is to be replaced by the given value,

a__ means the given value is added to the existing parameter value, and

r__ means the existing parameter value is multiplied by (1+ a given value).

Note: that there are always two underscores

<parname>	= SWAT parameter name (see the file).
<ext>	= SWAT file extension code for the file containing the parameter (see the file *Absolute_SWAT_Values.txt*)
<hydrogrp>	= (optional) soil hydrological group ('A','B','C' or 'D')
<soltext>	= (optional) soil texture
<landuse>	= (optional) name of the landuse category
<subbsn>	= (optional) subbasin number(s)

<slope>   = (optional) slope

Any combination of the above factors can be used to describe a parameter identifier. If the parameters are used globally, the identifiers <hydrogrp>, <soltext>, <landuse>, <subbsn>, and <slope> can be omitted.

Note: the two underscores for every previous specifications must be kept, i.e., to specify only the subbasin we must write   v__USLE_C.crp________2

The presented encoding scheme allows the user to make distributed parameters dependent on important influential factors such as: hydrological group, soil texture, landuse, and slope. The parameters can be kept regionally constant to modify a prior spatial pattern, or be changed globally. This gives the analyst larger freedom in selecting the complexity of a distributed parameter scheme. By using this flexibility, a calibration process can be started with a small number of parameters that only modify a given spatial pattern, with more complexity and regional resolution added in a stepwise learning process.

Specification of Soil Parameters

Parameter identifiers	Description
r__SOL_K(1).sol	K of Layer 1 of all HRUs
r__SOL_K(1,2,4-6).sol	K of Layer 1,2,4,5, and 6 of all HRUs
r__SOL_K().sol	K of All layers and all HRUs
r__SOL_K(1).sol__D	K of layer 1 of HRUs with hydrologic group D
r__SOL_K(1).sol____FSL	K of layer 1 of HRUs with soil texture FSL
r__SOL_K(1).sol____FSL__PAST	K of layer 1 of HRUs with soil texture FSL and landuse PAST
r__SOL_K(1).sol____FSL__PAST__1-3	K of layer 1 of subbasin 1,2, and 3 with HRUs containing soil texture FSL and landuse PAST

Specification of Management Parameters

Parameter identifiers	Description
v__HEAT_UNITS{rotation no,operation no}.mgt	Management parameters that are subject to operation/rotation must have both specified
v__CNOP{[],1}.mgt	This changes an operation's parameters in all rotations
v__CNOP{2,1,plant_id=33}.mgt	Changes CNOP for rotation 2, operation 1, and plant 33 only
v__CNOP{[],1,plant_id=33}.mgt	Similar to above, but for all rotations
v__CNOP{[],1,plant_id=33}.000010001.mgt	With this command you can only modify one file
r__manure_kg{9,1}.mgt	In these three examples, rotation 9, operation 1, and the rest are filters where , means AND
r__manure_kg{9,1,PLANT_ID=12}.mgt	
r__manure_kg{9,1,PLANT_ID=12,HUSC=0.15}.mgt	

Specification of Crop Parameters

Parameter identifiers	Description
v__T_OPT{30}.CROP.DAT	Parameter T_OPT for crop number 30 in the crop.dat file
v__PLTNFR(1){3}.CROP.DAT	Nitrogen uptake parameter #1 for crop number 3 in crop.dat file

Specification of Pesticide Parameters

Parameter identifiers	Description
v__WSOL{1}.pest.dat	This changes parameter WSOL for pesticide number 1 in pest.dat file

Specification of Precipitation Parameters

Parameter identifiers	Description
v__precipitation(1){1977300}.pcp1.pcp	(1) means column number 1 in the pcp file

	{1977300} specifies year and day
v__precipitation(1-3){1977300}.pcp1.pcp	(1-3) means column 1, 2, and3 {1977300} specifies year and day
v__precipitation( ){1977300,1977301}.pcp	( ) means all columns (all stations) {1977300,1977301} means 1977 days 300 and 301
v__precipitation( ){1977001-1977361,1978001-1978365,1979003}.pcp	( ) means all columns from day 1 to day 361 of 1977, and from day 1 to day 365 of 1978, and day 3 of 1979

Specification of slope Parameters

Parameter identifiers	Description
v__SOL_K(1).sol_______________0-10	K of layer 1 for HRUs with slope 0-10

Objective Function Definition

In observed.sf2 file, **Obj_Fn_Type(1=mult,2=sum,3=r2,4=chi2,5=NS,6=br2,7=ssrq)=** indicates the type of objective functions. Seven types of objective functions can be used:

- 1=mult A multiplicative form of the square error:

$$g = \frac{\sum_i (Q_m - Q_s)_i^2}{n_Q} * \frac{\sum_i (S_m - S_s)_i^2}{n_S} * \frac{\sum_i (N_m - N_s)_i^2}{n_N} * ....$$

Sometimes the denominator is divided by 1000 to keep g small.

2=sum A summation form of the square error:

$$g = w_1 \sum_{i=1}^{n_1} (Q_m - Q_s)_i^2 + w_2 \sum_{i=1}^{n_2} (S_m - S_s)_i^2 + w_3 \sum_{i=1}^{n_3} (N_m - N_s)_i^2 + ...$$

where weights w's could be calculated as:

i) $w_i = 1 / n_i \sigma_i^2$

where σ_i^2 is variance of the ith measured variable (see Abbaspour, et al., 2001), or

ii) $w_1 = 1, \quad w_2 = \overline{Q}_m / \overline{S}_m, \quad w_3 = \overline{Q}_m / \overline{N}_m$

where bars indicate averages (see Abbaspour et al., 1999). Note that choice of weighs can affect the outcome of an optimization exercise (see Abbaspour, et al., 1997).

3=r2 Coefficient of determination R^2 calculated as:

$$R^2 = \frac{\left[\sum_i (Q_{m,i} - \overline{Q}_m)(Q_{s,i} - \overline{Q}_s) \right]^2}{\sum_i (Q_{m,i} - \overline{Q}_m)^2 \sum_i (Q_{s,i} - \overline{Q}_s)^2}$$

If there is more than one variable, then the objective function is defined as:

$$g = \sum_i w_i R_i^2$$

4=Chi2 Chi-squared χ^2 calculated as:

$$\chi^2 = \frac{\sum_i (Q_m - Q_s)_i^2}{\sigma_Q^2}$$

If there is more than one variable, then the objective function is calculate as:

$$g = \sum_i w_i \chi_i^2$$

5=NS Nash-Sutcliffe (1970) coefficient calculated as:

$$NS = 1 - \frac{\sum_i (Q_m - Q_s)_i^2}{\sum_i \left(Q_{m,i} - \overline{Q}_m\right)^2}$$

If there is more than one variable, then the objective function is defined as:

$$g = \sum_i w_i NS_i$$

6=br2 Coefficient of determination R^2 multiplied by the coefficient of the regression line, bR^2. This function allows accounting for the discrepancy in the magnitude of two signals (depicted by b) as well as their dynamics (depicted by R^2). The objective function is expressed as:

$$\phi = \begin{cases} |b|R^2 & \text{if} \quad |b| \le 1 \\ |b|^{-1}R^2 & \text{if} \quad |b| > 1 \end{cases}$$

in case of multiple variables, g is defined as:

$$g = \sum_i w_i \phi_i$$

7=ssqr The SSQR method aims at the fitting of the frequency distributions of the observed and the simulated series. After independent ranking of the measured and the simulated values, new pairs are formed and the SSQR is calculated as

$$SSQR = \sum_{j=1,n} \left[Q_{j,measured} - Q_{j,simulated}\right]^2 \qquad (2)$$

where j represents the rank.

As opposed to the SSQ method, the time of occurrence of a given value of the variable is not accounted for in the SSQR method (van Griensven and Bauwens, 2003).

NOTE: After an iteration, one can change the type of objective function and run SUFI2_Post.bat alone to see the effect of different objective functions, without having to run SWAT again. This is quite informative as it shows how the choice of objective function affects the inverse solution.

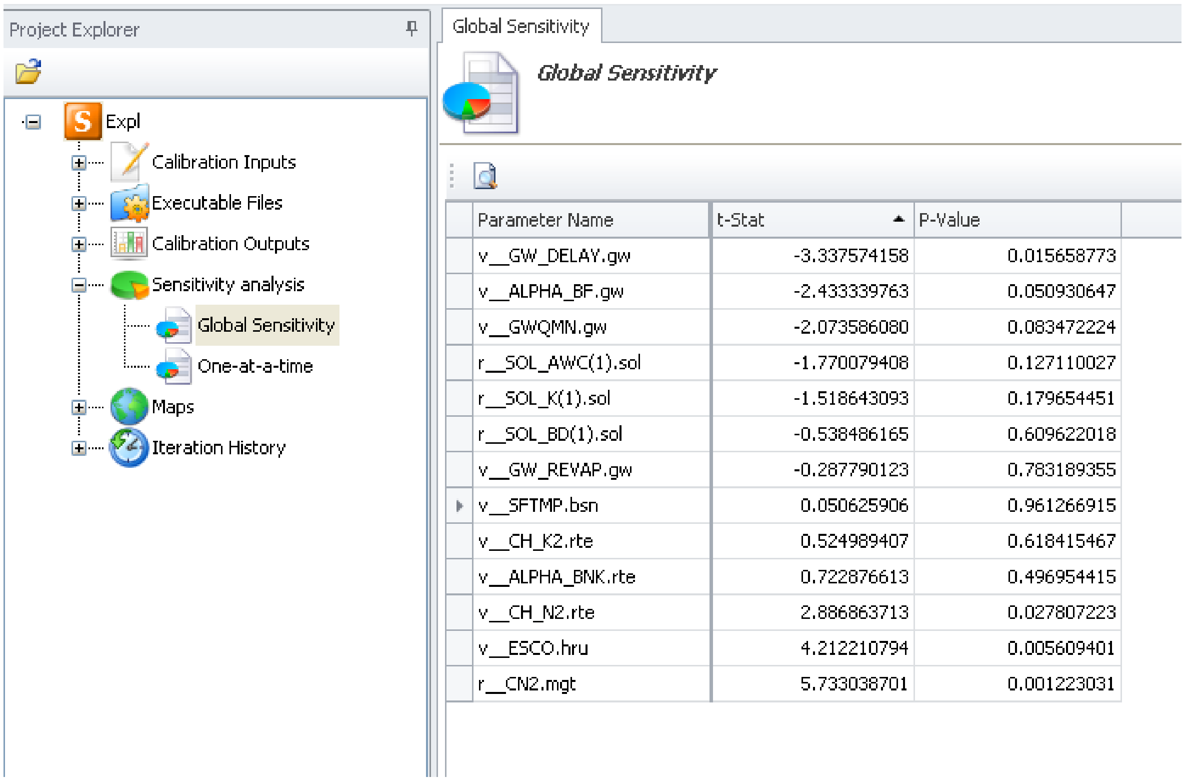
Sensitivity Analysis

1- Global Sensitivity analysis

Parameter sensitivities are determined by calculating the following multiple regression system, which regresses the Latin hypercube generated parameters against the objective function values (in file goal.sf2):

$$g = \alpha + \sum_{i=1}^{m} \beta_i b_i$$

A *t*-test is then used to identify the relative significance of each parameter b_i. The sensitivities given above are estimates of the average changes in the objective function resulting from changes in each parameter, while all other parameters are changing. This gives relative sensitivities based on linear approximations and, hence, only provides partial information about the sensitivity of the objective function to model parameters.

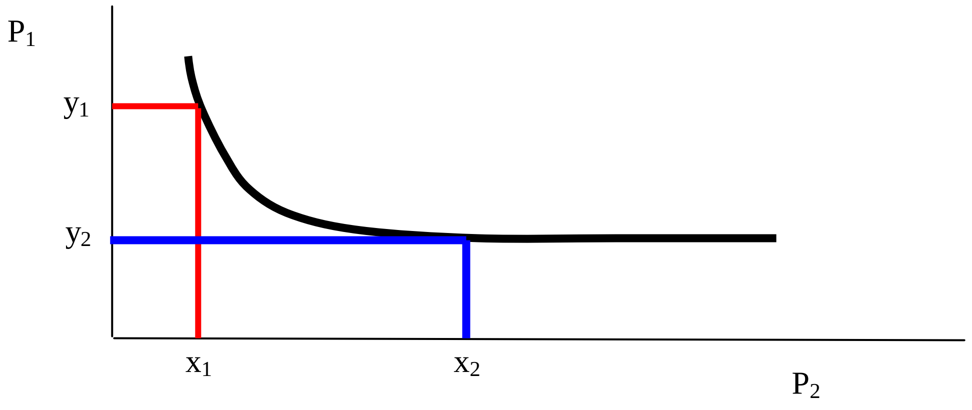


Parameter Name	t-Stat	P-Value
v__GW_DELAY.gw	-3.337574158	0.015658773
v__ALPHA_BF.gw	-2.433339763	0.050930647
v__GWQMN.gw	-2.073586080	0.083472224
r__SOL_AWC(1).sol	-1.770079408	0.127110027
r__SOL_K(1).sol	-1.518643093	0.179654451
r__SOL_BD(1).sol	-0.538486165	0.609622018
v__GW_REVAP.gw	-0.287790123	0.783189355
v__SFTMP.bsn	0.050625906	0.961266915
v__CH_K2.rte	0.524989407	0.618415467
v__ALPHA_BNK.rte	0.722876613	0.496954415
v__CH_N2.rte	2.886863713	0.027807223
v__ESCO.hru	4.212210794	0.005609401
r__CN2.mgt	5.733038701	0.001223031

t-stat provides a measure of sensitivity (larger in absolute values are more sensitive)
p-values determined the significance of the sensitivity. A values close to zero has more significance. In the above example, the most sensitive parameters are CN2 followed by ESCO and GW_DELAY.

2- One-at-a-time sensitivity analysis

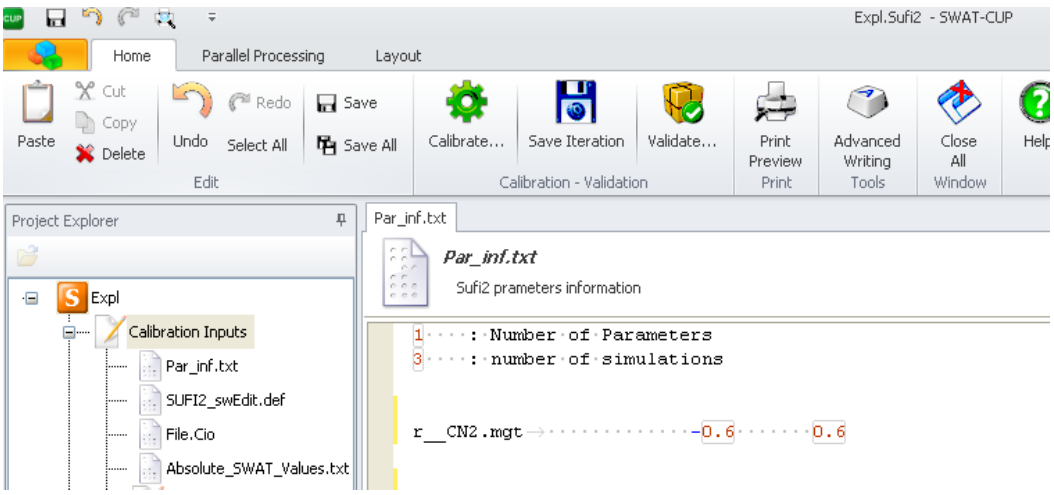
One-at-a-time sensitivity shows the sensitivity of a variable to the changes in a parameter if all other parameters are kept constant at some value. The problem here is that we never know what the value of those other constant parameters should be. This is an important consideration as the sensitivity of one parameter depends on the value of other parameters.



The above example illustrates this point. If value of parameter P_1 is kept constant at y_1, then small changes is parameter P_2 make significant changes in the objective function and indicate that P_2 is quite a sensitive parameter. While if the values of parameter P_1 is kept constant at y_2 value, then changes in parameters P_2 around x_2 will give the impression that P_2 is not a sensitive parameter. Therefore, the values of the fixed parameters make a difference to the sensitivity of a parameter.

To perform the one-at-a-time sensitivity analysis:

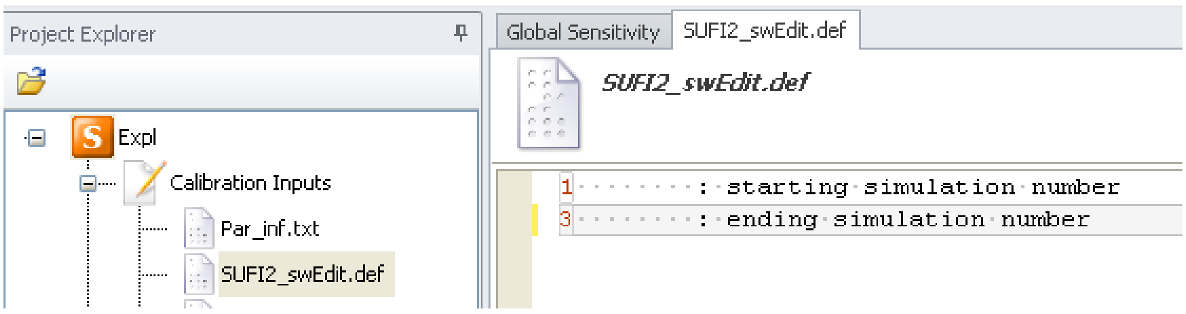
1- Do as shown in the following Figure. Set the number of parameters in the Par_:inf.txt file to 1, and perform a minimum of 3 simulations.



Par_inf.txt

```
1    : Number of Parameters
3    : number of simulations


r__CN2.mgt              -0.6        0.6
```

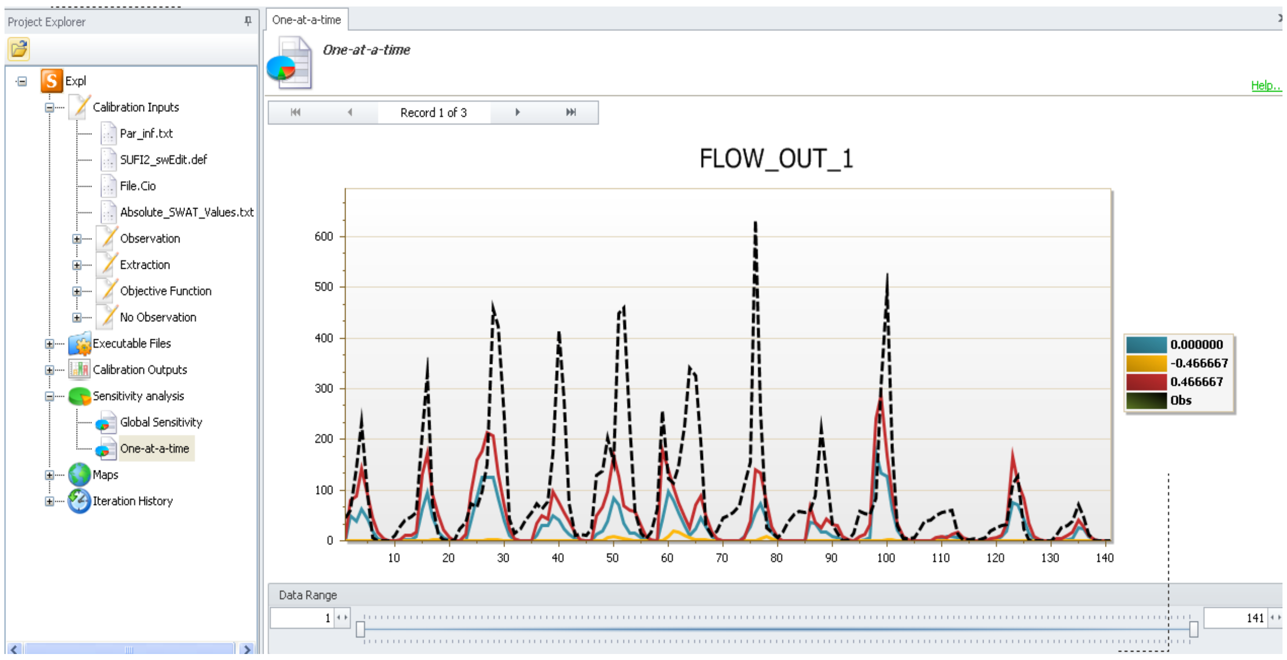
2- Then set the values of file SUFI2_swEdit.def as follows:



```
1        : starting simulation number
3        : ending simulation number
```

3- Finally perform the simulation by running under **Calibration -----> SUFI2_pre.bat** and then **SUFI2_run.bat.**

4- Now, the three simulation can be visualized for each variable by executing **one-at-a-time** command under **Sensitivity analysis as** shown below:



The dashed line is the observation and the discharge signal for FLOW_OUT_1 is plotted for three values of CN2 within the specified range. Clearly, CN2 needs to have larger values.

NOTE: The users must be aware that the parameters in the SWAT files in the main project directory are always changing. Once you perform an iteration, then the parameter values in those files are the values of the last run (last parameter set) of the last iteration. To perform the one-at-a-time sensitivity analysis, one should set the values of the parameters that are kept constant to some reasonable values. These reasonable values could, for example, be the best simulation (simulation with the best objective function value) of the last iteration.

To set the parameters to the best value of the last iteration,

1- Note the number of the best simulation in the Summary_Stat.txt file

2- In the SUFI2_swEdit.txt set the starting and ending simulation values to the number of the best simulation.

3- Under Calibration, run SUFI2_run.bat

This command will replace the parameter values and set them to the best values of the last iteration. If there is no need to run SWAT here, then the program can be stopped when SWAT run begins.

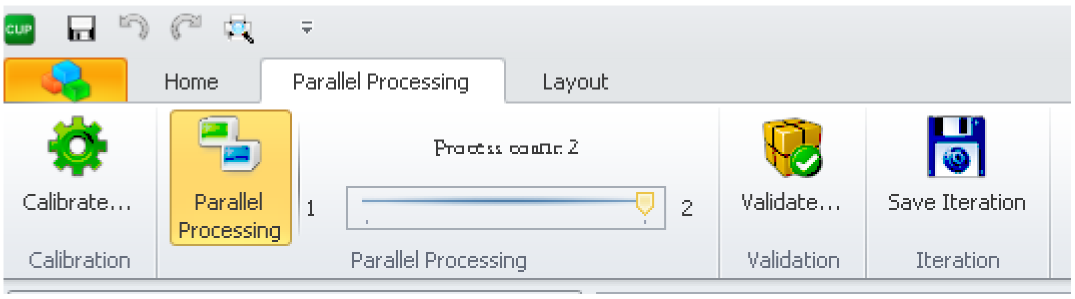
Parallel Processing

Parallel processing is a licensed product. Its function is to speed up the calibration process by parallelizing the runs in SUFI2. A procedure is being worked out for PSO also. The speed of the parallel processing depends on the characteristics of the computer. New laptops now have at least 4 CPUs. The parallel processing module can utilize all 4 CPUs are so a 1000 run iteration can be divided into 4 simultaneous runs of 250 each per CPU. The speedup will not be 4 times because of program and Windows overheads; but the run with parallel processing will be substantially faster than a single run submission.

Now a days it is possible to build quite inexpensively a computer with 48 to 64 CPUs and more than 24 GB of RAM. Most SWAT models of any detail could be run on such machines without the need for cloud or grid computing.

Currently, 20 simulations are allowed to be made without the need for a license. To activate parallel processing, simply click the **Parallel Processing** button on the command bar. A new set of command icons appear. Press **Parallel Processing** icon again. This will show the number of parallel processes that can be submitted to the computer in use. If the size of the project is large and there is not enough memory, then smaller number of parallel processes than the number of CPUs may be possible. The Calibration icon here works as before.

A paper submitted to Environmental Modelling and Software (Rouholahnejad, et al., 2011) could be consulted for more details.



The sequence of program execution

The sequence of program execution and input/outputs are shown in Figure 13. In the following, each input and output file is described in detail.

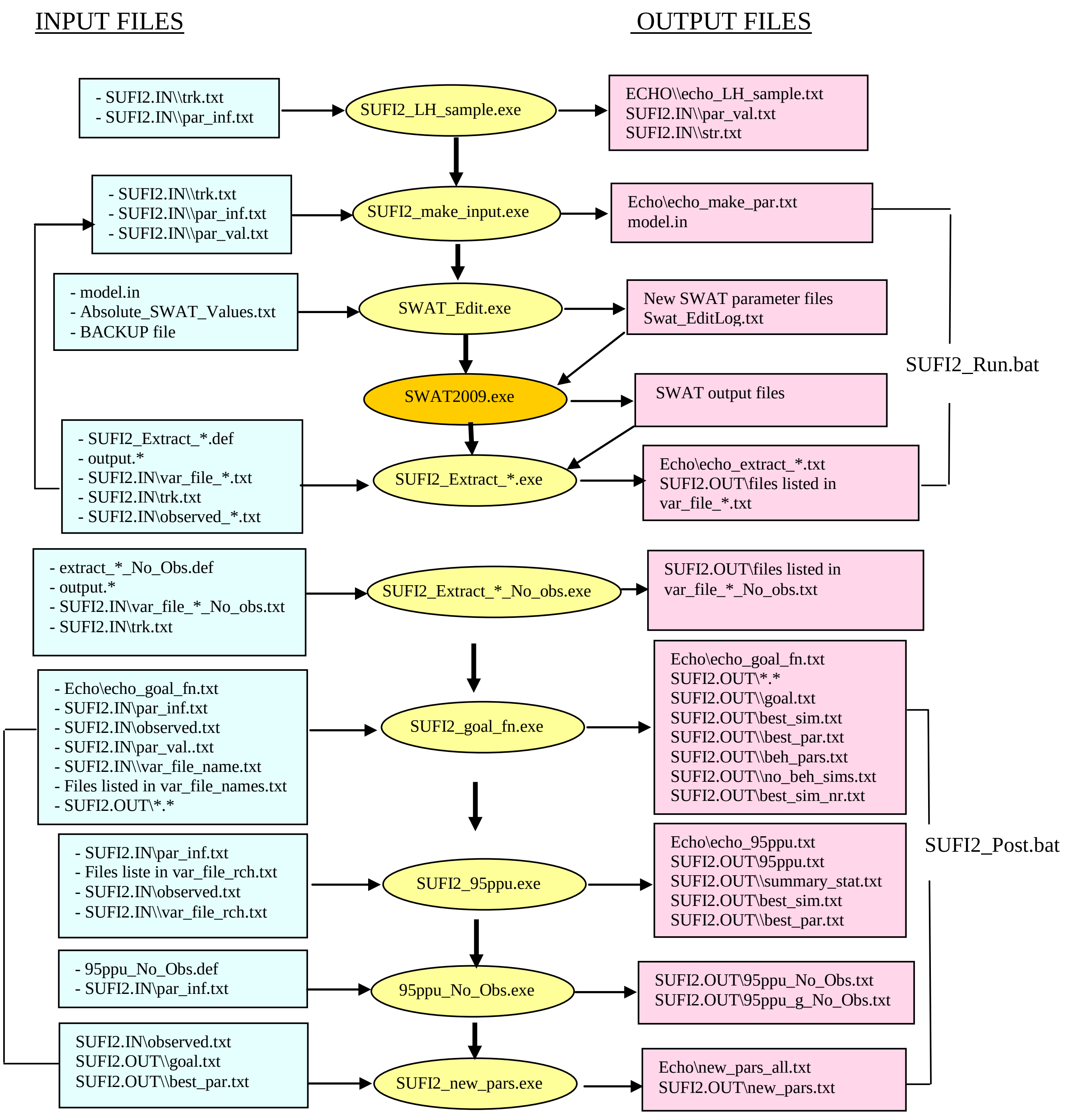


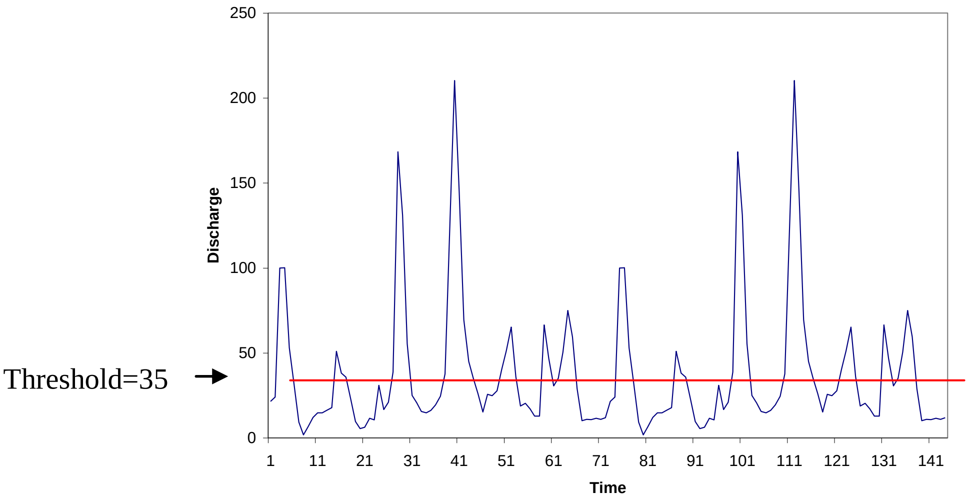
Figure 13. Sequence of program execution and input/output files in SUFI2

FILE DEFINITION

Parameter in Observed.txt

- **var_weight=** is the weight of each variable, i.e., discharge, sediment concentration etc. This weight could be chosen such that contribution of each variable to the objective function is equal as explained above. In file \Echo\echo_goal_fn.sf2 at the last line contribution of each variable to the objective function is given. Based on this one can adjust the weights as desired and run the SUFI2_Post.bat again.

- **var_Threshold=** is a threshold where a signal is divided into two parts. We refer to this as a "multi-component" assignment (see Abbaspour et al., 2004). Values smaller than the threshold and values larger than the threshold are treated as two variables. This is to ensure that, for example, base flow has the same values as the peak flows. If you choose option 2 for objective function, i.e., mean square error, then base flow my not have much effect on the optimization, hence, peak flow will dominate the processes. With this option they can be given the same weight. This option is most effective for option 2 of objective function and is not defined for R^2 and bR^2.



In case multi-component assignment is used, all objective functions above are divided into a lower and an upper part. To not use this option, simply set *var-threshold* to a negative number (say -1 for a variable that is always positive) and the weights for upper and lower thresholds to 1.

- wt_below_threshold and **wt_above_threshold** are the weights for the two components.In file \Echo\echo_goal_fn.sf2 at the last line contribution of each variable for lower and upper section of the variable is given. Based on this one can adjust the weights as desired and run the SUFI2_Post.bat again. See the definition of objective functions and weights above.

- pcnt_Error= is the percentage of error in the measurement. This is used in the calculation of the percentage of data bracketed by the 95% prediction uncertainty

- no_obs= this indicated the number of observed data for each variable.

The above format is repeated for every variable in the objective function.

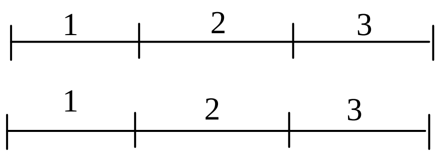
Latin Hypercube Sampling

SUFI2_pre.bat

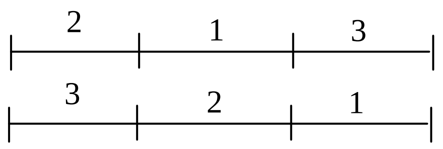
The batch file *SUFI2_pre.bat* runs the *SUFI2_LH_sample.exe* program, which generates Latin hypercube samples. These samples are stored in *par_val.sf2* file.

This program uses Latin hypercube sampling to sample from the parameter intervals given in *par_inf.sf2* file. The sampled parameters are given in *par_val.sf2* file, while the structure of the sampled data is written to *str.sf2* just for information. If the number of simulations is 3, then the following happens:

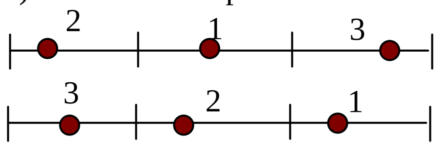
1) Parameters (say 2) are divided into the indicated number of simulations (say 3)



2) Parameter segments are randomized



3) A random sample is taken in every segment



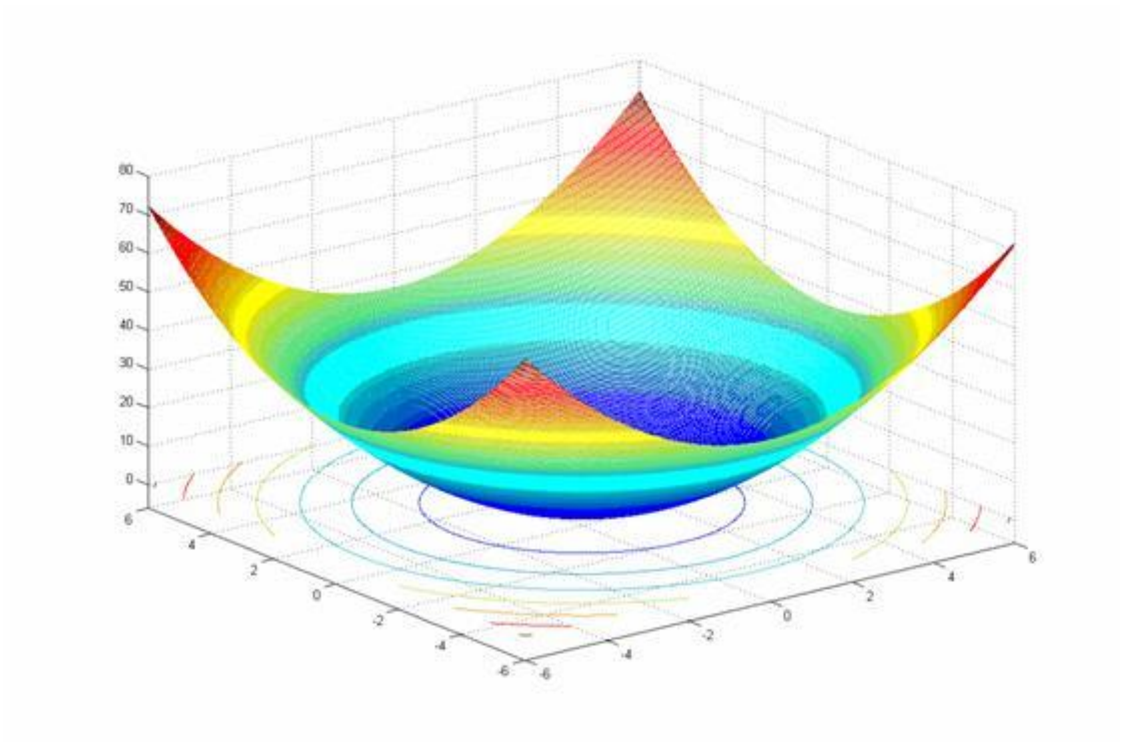
Every vertical combination is then a parameter set.

Validation in SUFI2

To perform validation in SUFI2, edit the files observed_rch.txt, observed_hru.txt, obsrved_sub.txt, and observed.txt as necessary for the validation period. Also, the extraction files and the file.cio to reflect the validation period. Then simply use the calibrated parameter ranges to make one complete iteration without changing the parameters further.

PSO

Particle Swarm Optimization



1. Introduction

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. The detailed information will be given in following sections.

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

The remaining of the report includes six sections:

Background: artificial life.

The Algorithm

Comparisons between Genetic algorithm and PSO

Artificial neural network and PSO

PSO parameter control

Online resources of PSO

2. Background: Artificial life

The term "Artificial Life" (ALife) is used to describe research into human-made systems that possess some of the essential properties of life. ALife includes two-folded research topic: (http://www.alife.org)

1. ALife studies how computational techniques can help when studying biological phenomena

2. ALife studies how biological techniques can help out with computational problems

The focus of this report is on the second topic. Actually, there are already lots of computational techniques inspired by biological systems. For example, artificial neural network is a simplified model of human brain; genetic algorithm is inspired by the human evolution.

Here we discuss another type of biological system - social system, more specifically, the collective behaviors of simple individuals interacting with their environment and each other. Someone called it as swarm intelligence. All of the simulations utilized local processes, such as those modeled by cellular automata, and might underlie the unpredictable group dynamics of social behavior.

Some popular examples are floys and boids. Both of the simulations were created to interpret the movement of organisms in a bird flock or fish school. These simulations are normally used in computer animation or computer aided design.

There are two popular swarm inspired methods in computational intelligence areas: Ant colony optimization (ACO) and particle swarm optimization (PSO). ACO was inspired by the behaviors of ants and has many successful applications in discrete optimization problems. (http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html)

The particle swarm concept originated as a simulation of simplified social system. The original intent was to graphically simulate the choreography of bird of a bird block or fish school. However, it was found that particle swarm model can be used as an optimizer. (http://www.engr.iupui.edu/~shi/Coference/psopap4.html)

3. The algorithm

As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

PSO learns from the scenario and uses it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

v[ ] = v[ ] + c1 * rand() * (pbest[ ] - present[ ]) + c2 * rand() * (gbest[ ] - present[ ]) (a)

present[] = persent[] + v[ ] (b)

v[ ] is the particle velocity, persent[ ] is the current particle (solution). pbest[ ] and gbest[ ] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2.

The pseudo code of the procedure is as follows

```
For each particle
    Initialize particle
END
Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest) in history
```

```
                    set current value as the new pBest
        End
        Choose the particle with the best fitness value of all the particles as the gBest
        For each particle
                Calculate particle velocity according equation (a)
                Update particle position according equation (b)
End
While maximum iterations or minimum error criteria is not attained
```

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.

4. Comparisons between Genetic Algorithm and PSO

Most of evolutionary techniques have the following procedure:

1. Random generation of an initial population

2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.

3. Reproduction of the population based on fitness values.

4. If requirements are met, then stop. Otherwise go back to 2.

From the procedure, we can learn that PSO shares many common points with GA. Both algorithms start with a group of a randomly generated population, both have fitness values to evaluate the population. Both update the population and search for the optimum with random techniques. Both systems do not guarantee success.

However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm.

Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gBest (or lBest) gives out the information to others. It is a one-way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

5. Artificial neural network and PSO

An artificial neural network (ANN) is an analysis paradigm that is a simple model of the brain and the back-propagation algorithm is the one of the most popular method to train the artificial neural network. Recently there have been significant research efforts to apply evolutionary computation (EC) techniques for the purposes of evolving one or more aspects of artificial neural networks.

Evolutionary computation methodologies have been applied to three main attributes of neural networks: network connection weights, network architecture (network topology, transfer function), and network learning algorithms.

Most of the work involving the evolution of ANN has focused on the network weights and topological structure. Usually the weights and/or topological structure are encoded as a chromosome in GA. The selection of fitness function depends on the research goals. For a classification problem, the rate of mis-classified patterns can be viewed as the fitness value.

The advantage of the EC is that EC can be used in cases with non-differentiable PE transfer functions and no gradient information available. The disadvantages are 1. The performance is not competitive in some problems. 2. representation of the weights is difficult and the genetic operators have to be carefully selected or developed.

There are several papers reported using PSO to replace the back-propagation learning algorithm in ANN in the past several years. It showed PSO is a promising method to train ANN. It is faster and gets better results in most cases. It also avoids some of the problems GA met.

Here we show a simple example of evolving ANN with PSO. The problem is a benchmark function of classification problem: iris data set. Measurements of four attributes of iris flowers are provided in each data set record: sepal length, sepal width, petal length, and petal width. Fifty sets of measurements are present for each of three varieties of iris flowers, for a total of 150 records, or patterns.

A 3-layer ANN is used to do the classification. There are 4 inputs and 3 outputs. So the input layer has 4 neurons and the output layer has 3 neurons. One can evolve the number of hidden neurons. However, for demonstration only, here we suppose the hidden layer has 6 neurons. We can evolve other parameters in the feed-forward network. Here we only evolve the network weights. So the particle will be a group of weights, there are 4*6+6*3 = 42 weights, so the particle consists of 42 real numbers. The range of weights can be set to [-100, 100] (this is just a example, in real cases, one might try different ranges). After encoding the particles, we need to determine the fitness function. For the classification problem, we feed all the patterns to the network whose weights is determined by the particle, get the outputs and compare it the standard outputs. Then we record the number of misclassified patterns as the fitness value of that particle. Now we can apply PSO to train the ANN to get lower number of misclassified patterns as possible. There are not many parameters in PSO need to be adjusted. We only need to adjust the number of hidden layers and the range of the weights to get better results in different trials.

6. PSO parameter control

There are not too many parameters needing to be tuned in PSO. Here is a list of the parameters and their typical values.

The number of particles: the typical range is 20 - 40. Actually for most of the problems 10 particles is large enough to get good results. For some difficult or special problems, one can try 100 or 200 particles as well.

Dimension of particles: It is determined by the problem to be optimized.

Range of particles: It is also determined by the problem to be optimized, you can specify different ranges for different dimension of particles.

Vmax: it determines the maximum change one particle can take during one iteration. Usually we set the range of the particle as the Vmax for example, the particle (x_1, x_2, x_3)

X_1 belongs [-10, 10], then Vmax = 20

Learning factors: c1 and c2 usually equal to 2. However, other settings were also used in different papers. But usually c1 equals to c2 and ranges from [0, 4]

The stop condition: the maximum number of iterations the PSO execute and the minimum error requirement. For example, for ANN training in previous section, we can set the minimum error requirement is one misclassified pattern. The maximum number of iterations is set to 2000. This stop condition depends on the problem to be optimized.

7. Online Resources of PSO

The development of PSO is still ongoing. And there are still many unknown areas in PSO research such as the mathematical validation of particle swarm theory.

One can find much information from the internet. Following are some information you can get online:

http://www.particleswarm.net lots of information about Particle Swarms and, particularly, Particle Swarm Optimization. Lots of Particle Swarm Links.

http://icdweb.cc.purdue.edu/~hux/PSO.shtml lists an updated bibliography of particle swarm optimization and some online paper links

http://www.researchindex.com/ you can search particle swarm related papers and references.

References:

http://www.engr.iupui.edu/~eberhart/

http://users.erols.com/cathyk/jimk.html

http://www.alife.org

http://www.aridolan.com

http://www.red3d.com/cwr/boids/

http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html

http://www.engr.iupui.edu/~shi/Coference/psopap4.html

Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proc. IEEE int'l conf. on neural networks Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.

Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan, 1995.

Eberhart, R. C. and Shi, Y. Particle swarm optimization: developments, applications and resources. Proc. congress on evolutionary computation 2001 IEEE service center, Piscataway, NJ., Seoul, Korea., 2001.

Eberhart, R. C. and Shi, Y. Evolving artificial neural networks. Proc. 1998 Int'l Conf. on neural networks and brain pp. PL5-PL13. Beijing, P. R. China, 1998.

Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization. Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf., Springer-Verlag, Berlin, San Diego, CA., 1998.

Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization. Evolutionary Programming VII: Proc. EP 98 pp. 591-600. Springer-Verlag, New York, 1998.

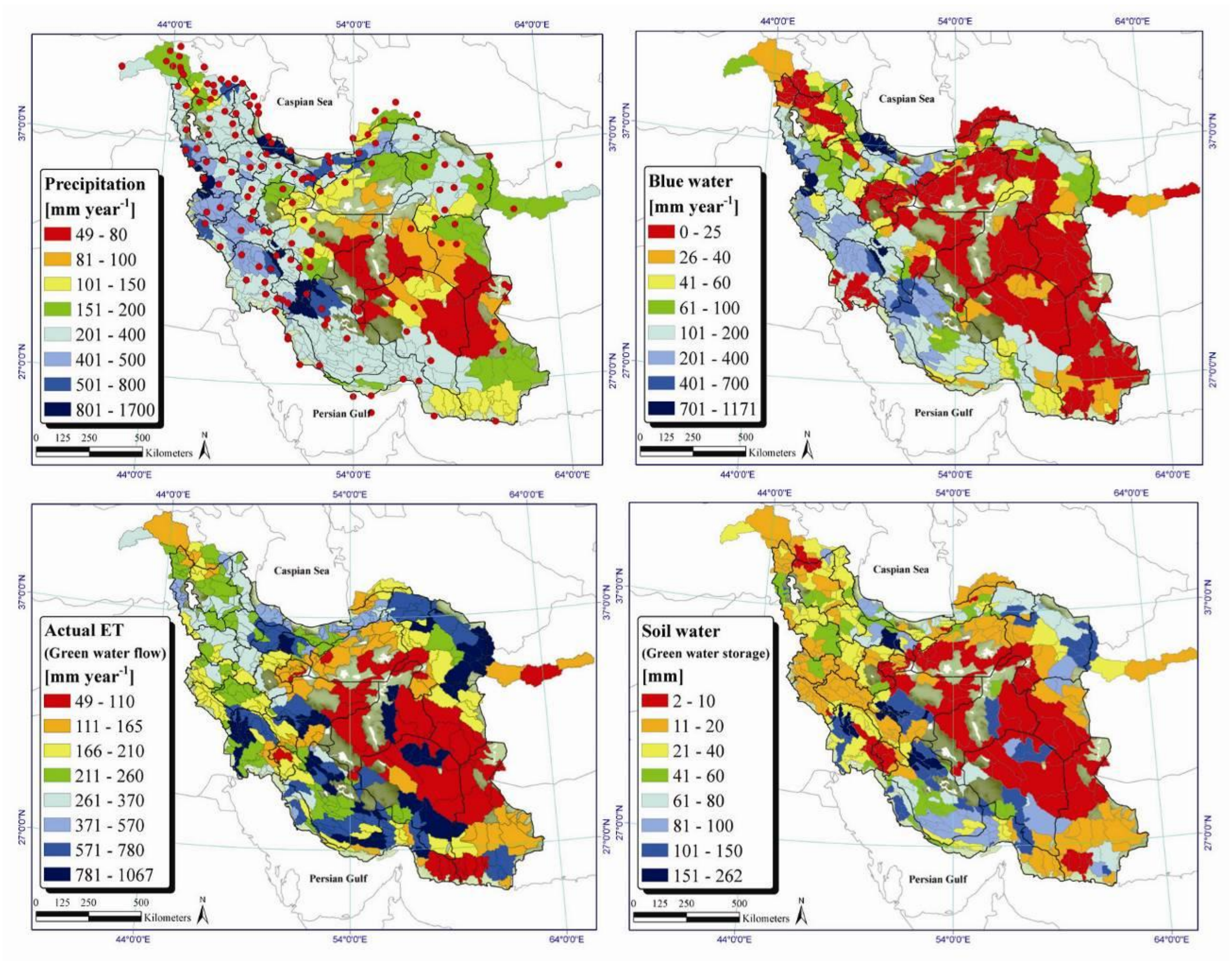
Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation pp. 69-73. IEEE Press, Piscataway, NJ, 1998

Source of this article is:

http://www.swarmintelligence.org/

GLUE

Generalized Likelihood Uncertainty Estimation



Introduction to the Program GLUE

A short summary of the GLUE (Beven and Binley, 1992) concept is given below. For more information the readers are referred to the GLUE literature and the Internet.

The Generalized Likelihood Uncertainty Estimation (GLUE) (Beven and Binley, 1992) was introduced partly to allow for the possible non-uniqueness (or equifinality) of parameter sets during the estimation of model parameters in over-parameterized models. The procedure is simple and requires few assumptions when used in practical applications. GLUE assumes that, in the case of large over-parameterized models, there is no unique set of parameters, which optimizes goodness-of fit-criteria. The technique is based on the estimation of the weights or probabilities associated with different parameter sets, based on the use of a subjective likelihood measure to derive a posterior probability function, which is subsequently used to derive the predictive probability of the output variables. In Romanowicz et al., (1994) a statistically motivated, more formal equivalent of GLUE is developed, where the likelihood function is explicitly derived based on the error between the observed outputs and those simulated by the model. This formal approach is equivalent to a Bayesian statistical estimation: it requires assumptions about the statistical structure of the errors. GLUE is usually applied by directly likelihood weighting the outputs of multiple model realizations (deterministic or stochastic, defined by sets of parameter values within one or more model structures) to form a predictive distribution of a variable of interest. Prediction uncertainties are then related to variation in model outputs, without necessarily adding an additional explicit error component. There is thus an interesting question as to whether an appropriate choice of likelihood measure can produce similar results from the two approaches.

There are a number of possible measures of model performance that can be used in this kind of analysis. The only formal requirements for use in a GLUE analysis are that the likelihood measure should increase monotonously with increasing performance and be zero for models considered as unacceptable or non-behavioral. Application-oriented measures are easily used in this framework. Measures based on formal statistical assumptions, when applied to all model realizations (rather than simply in the region of an "optimal" model) should give results similar to a Bayesian approach when used within a GLUE framework (Romanowicz et al., 1994), but the assumptions made (additive Gaussian errors in the simplest cases) are not always easily justified in the case of nonlinear environmental models with poorly known boundary conditions.

A GLUE analysis consists of the following three steps:

1) After the definition of the "generalized likelihood measure" $L(\theta)$, a large number of parameter sets are randomly sampled from the prior distribution and each parameter set is assessed as either "behavioral" or "non-behavioral" through a comparison of the "likelihood measure" with the given threshold value.

2) Each behavioral parameter is given a "likelihood weight" according to:

$$w_i = \frac{L(\theta_i)}{\sum_{k=1}^{N} L(\theta_k)} \tag{1}$$

where N is the number of behavioral parameter sets.

3) Finally, the prediction uncertainty is described as prediction quantile from the cumulative distribution realized from the weighted behavioral parameter sets.

In literature, the most frequently used likelihood measure for GLUE is the *Nash-Sutcliffe* coefficient (*NS*), which is also used in the GLUE06 program:

$$NS = 1 - \frac{\sum_{t_i=1}^{n} (y_{t_i}^{M}(\boldsymbol{\theta}) - y_{t_i})^2}{\sum_{t_i=1}^{n} (y_{t_i} - \bar{y})^2} \tag{2}$$

where n is the number of the observed data points, and y_{t_i} and $y_{t_i}^{M}(\boldsymbol{\theta})$ represents the observation and model simulation with parameter θ at time t_i, respectively, and $\bar{y}$ is the average value of the observations.

Coupling of GLUE to SWAT-CUP

SWAT-CUP is an interface to facilitate the coupling between external system analysis tools and SWAT model. The following diagram illustrates the GLUE-SWAT-CUP links.

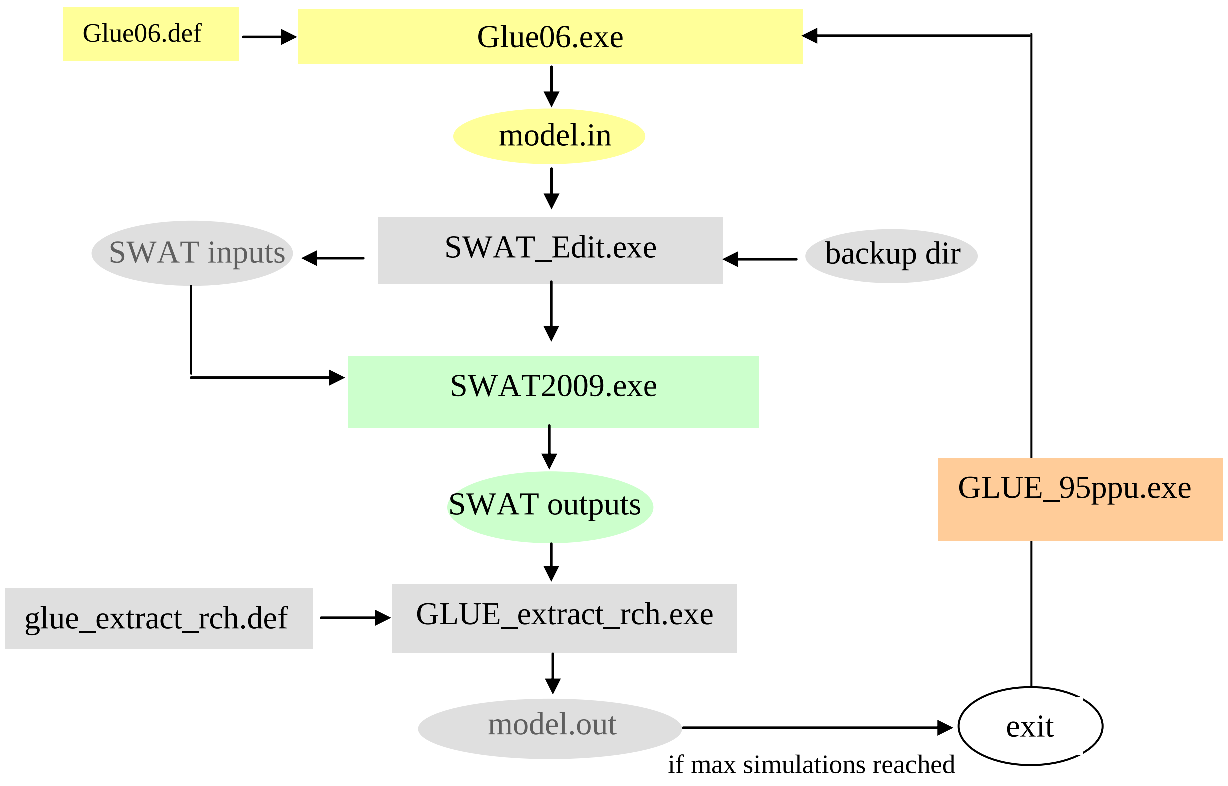
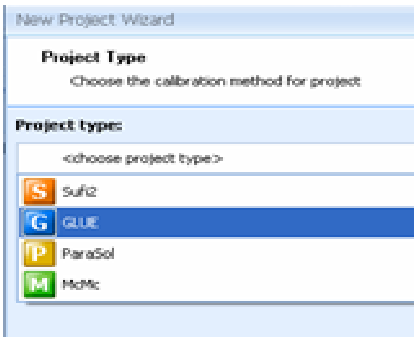


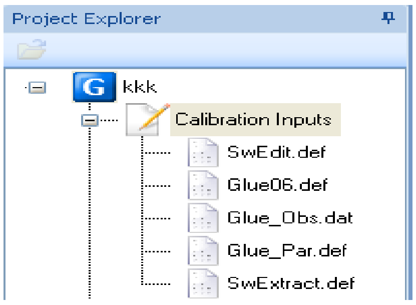
Figure 14. Interface of GLUE and SWAT-CUP.

Step-by-step procedure to run GLUE in SWAT-CUP

1) Follow the initial steps as in SUFI-2, but choose a GLUE project type

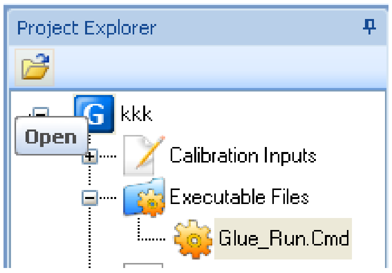


2) Edit the files in "Calibration Input"

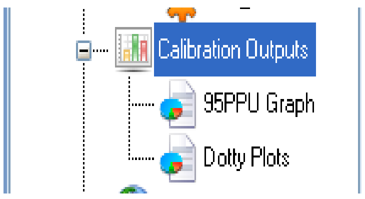


3) Edit the command file under "Execution Files" if necessary

4) Execute the program by running Glue06.exe and then Glue_95ppu.exe.



5) Examine the output in the "Calibration Output". For Glue a large number of simulations (>5000) is usually required. Convergence may need to be confirmed by making a run of a larger number of simulations and comparing the objective functions, the dotty plots, and the 95PPU. Further processing can be done on GLUE outputs as required by the user.



In Figure 15, the execution order and each input and output file of GLUE is listed. The entries are self explanatory.

model.in
GLUE.OUT\modelpara.out
GLUE.OUT\modelres.out
GLUE.OUT\modelpara.beh
GLUE.OUT\modelquant.out
GLUE.OUT\modelres.beh

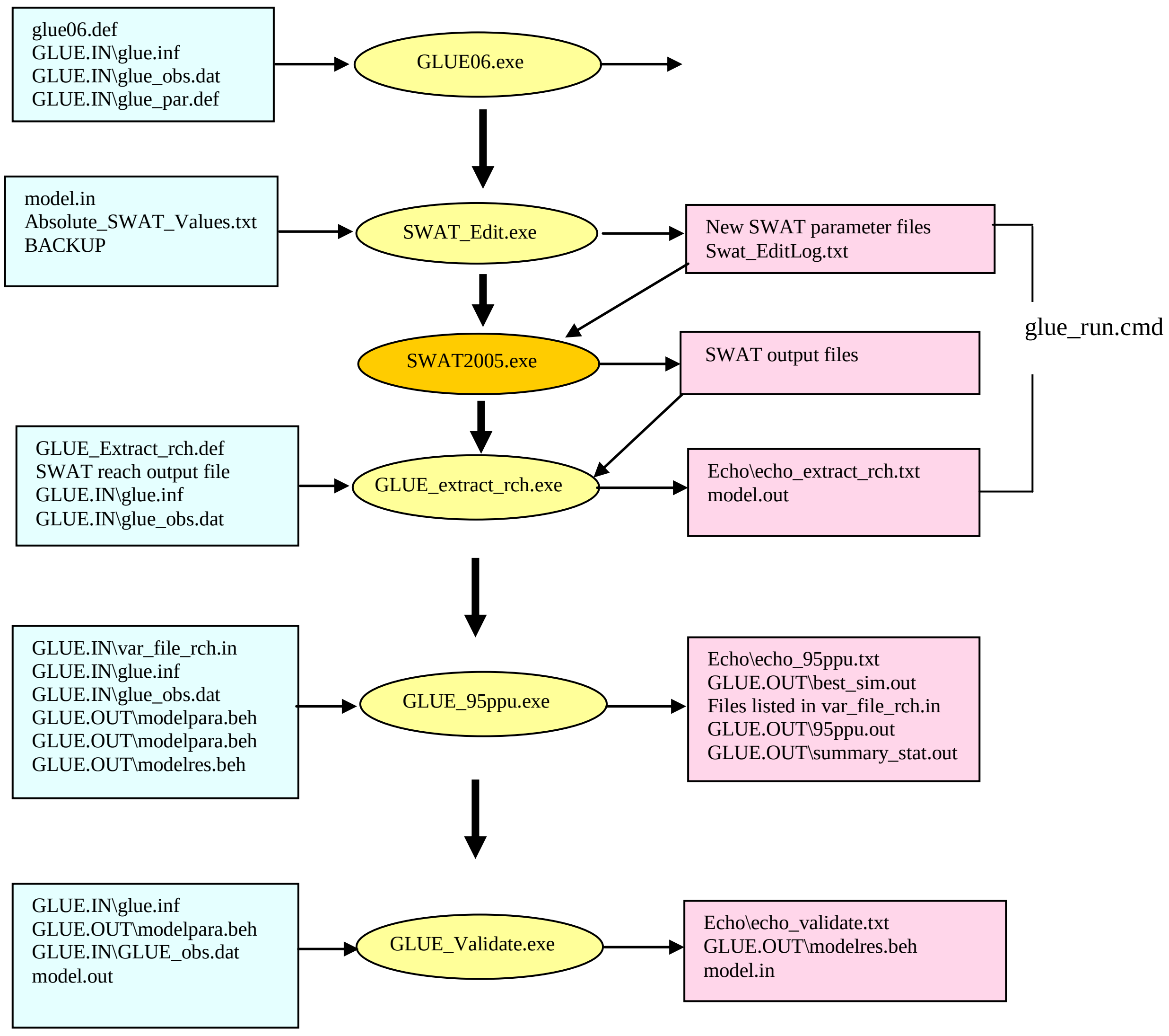


Figure 15. Sequence of program execution and input/output files in GLUE

Outputs of Glue are in the following files:

modelpara.beh	Contains the behavioural parameter sets as well as the value of the objective function
modelpara.out	Contains all parameter sets as wll as the value of the objective function
modelquant.out	Contains the 95% prediction uncertainty (95PPU) of output variables
modelres.beh	Contains model simulation for behavioural parameters
modelres.out	Contains all model simulations

VALIDATION

After calibration, validation can be performed by using the “Validate” option from the menu. Before executing validation, however, the GLUE_obs.dat file must be edited to contain validation data, GLUE_Extract_rch.def must be edited to extract validation data, and SWAT’s File.cio and climate files (pcp.pcp etc.) must cover the validation period. The validation program uses the good parameters only to run SWAT.

Input files of GLUE are described below. They are for most parts self explanatory.

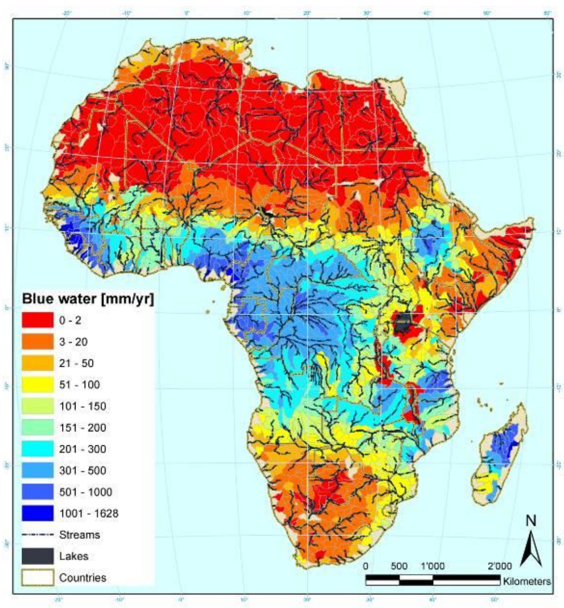
File Definition

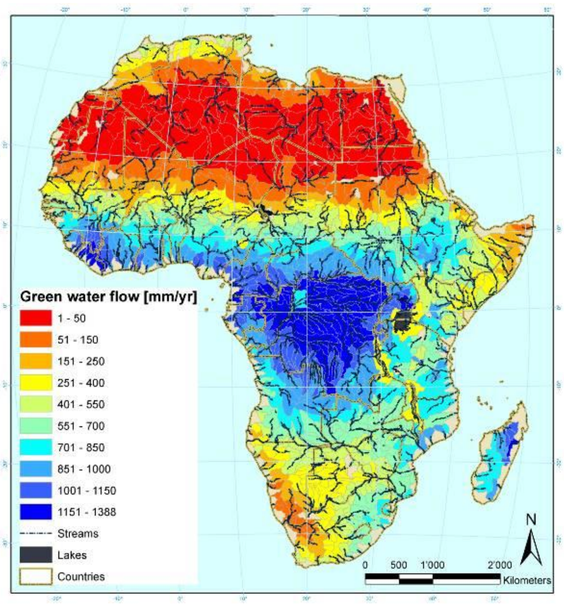
glue06.def

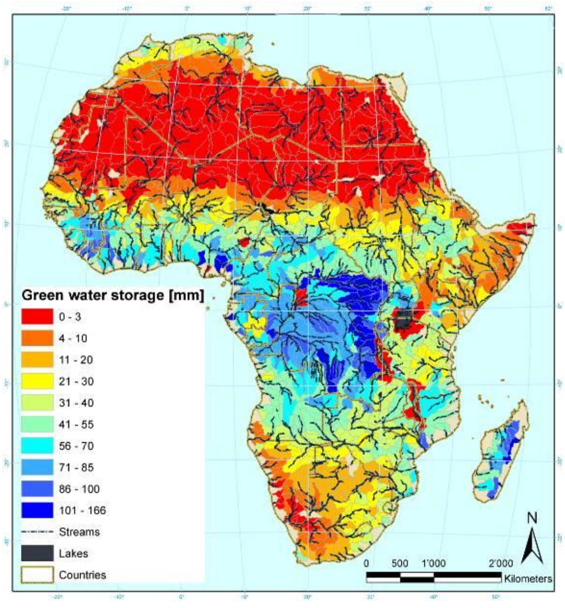
Line	parameter		value	Remark
1	// comment			
2	MaxSimulation		10000	The larger, the better!
3	ParDefFile		glue_par.def	parameter definition file
4	ObjfunThresh		0.3	Threshold value given by the user to separate the behavioural parameters from the non-behavioural parameters
5	Percentile		0.025	The percentile used to calculate the quantiles of behavioural model results in line 14
6	ModelInFile		model.in	output of glue06.exe, and the input of SWAT_Edit.exe
7	ModelOutFile		model.out	output of GLUE_extract_rch.exe and input of glue06.exe
8	ModelCmd		glue_run.cmd	Bach file executed during GLUE run
9	ModelObjfunFile	F	glue_obs.dat	If the first parameter is “F”, then the second parameter is the observed data filename and Nash-Sutcliffe is the objective function. If the first parameter is “T”, then the second parameter is the objective function filename that must be calculated and provided by the user
10	ModelParaSet		modelpara.out	The output filename for all sampled parameter sets
11	ModelBehParaset		modelpara.beh	The output filename for the behavioural parameter sets
12	ModelResult	T	modelres.out	The output filename for all the model results
13	ModelBehResult		modelres.beh	The output filename for the behavioural model results
14	ModelResQaunt	T	modelquant.out	The output filename for the quantiles of behavioural model results

ParaSol

Parameter Solution







Introduction to the Program ParaSol

A short summary of the ParaSol (Van Griensven and Meixner, 2006) concept is given below. For more information the readers are referred to the APPENDIX, the literature and the Internet.

The ParaSol method aggregates objective functions (OF's) into a global optimization criterion (GOC), minimizes these OF's or a GOC using the Shuffle Complex (SCE-UA) algorithm and performs uncertainty analysis with a choice between 2 statistical concepts. The SCE algorithm is a global search algorithm for the minimization of a single function for up to 16 parameters (*Duan et al*., 1992). It combines the direct search method of the simplex procedure with the concept of a controlled random search of Nelder and Mead (1965), a systematic evolution of points in the direction of global improvement, competitive evolution (Holland, 1975) and the concept of complex shuffling. In a first step (zero-loop), SCE-UA selects an initial 'population' by random sampling throughout the feasible parameters space for *p* parameters to be optimized (delineated by given parameter ranges). The population is portioned into several "complexes" that consist of 2*p*+1 points. Each complex evolves independently using the simplex algorithm. The complexes are periodically shuffled to form new complexes in order to share information between the complexes.

SCE-UA has been widely used in watershed model calibration and other areas of hydrology such as soil erosion, subsurface hydrology, remote sensing and land surface modeling (Duan, 2003). It was generally found to be robust, effective and efficient (Duan, 2003). The SCE-UA has also been applied with success on SWAT for the hydrologic parameters (Eckardt and Arnold, 2001) and hydrologic and water quality parameters (van Griensven and Bauwens, 2006). The procedure of ParaSol is:

1) After the optimization of the modified SCE-UA, the simulations performed are divided into 'good' simulations and 'not good' simulations by a threshold in this way similar to the GLUE methodology, and accordingly, 'good' parameter sets and 'not good' parameter set. Unlike GLUE, the threshold value can be defined by either the χ^2-statistics where the selected simulations correspond to the confidence region (CR) or Bayesian statistics that are able to point out the high probability density region (HPD) for the parameters or the model outputs.

2) The prediction uncertainty is hence constructed equally from the 'good' simulations.

The Objective function used in ParaSol is *Sum of the squares of the residuals (SSQ)*:

$$SSQ = \sum_{t_i=1}^{n} (y_{t_i}^{M}(\boldsymbol{\theta}) - y_{t_i})^2 \qquad (3)$$

Coupling ParaSol to SWAT-CUP

The dataflow between program ParaSol and SWAT-CUP is as shown below.

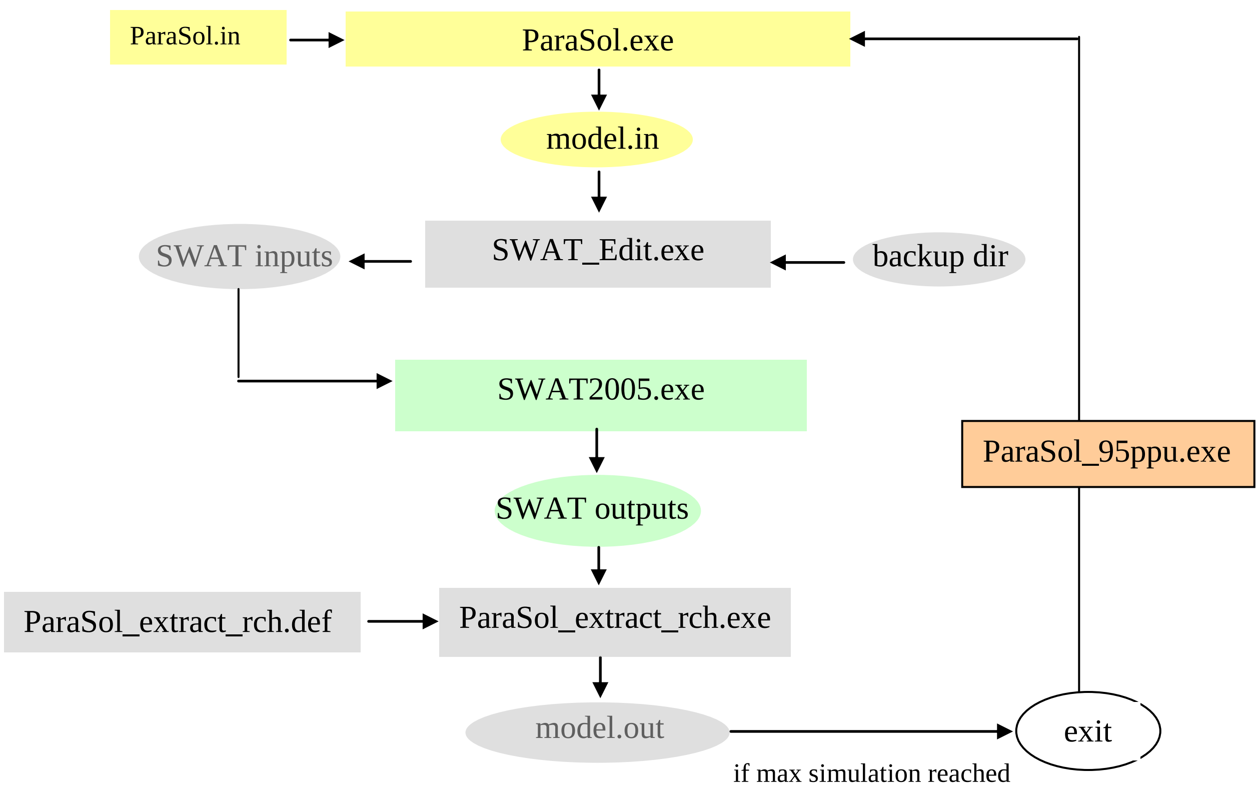
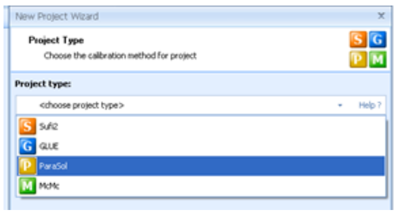


Figure 16. Interface of ParaSol and SWAT-CUP.

Step-by-step procedure to run ParaSol in SWAT-CUP

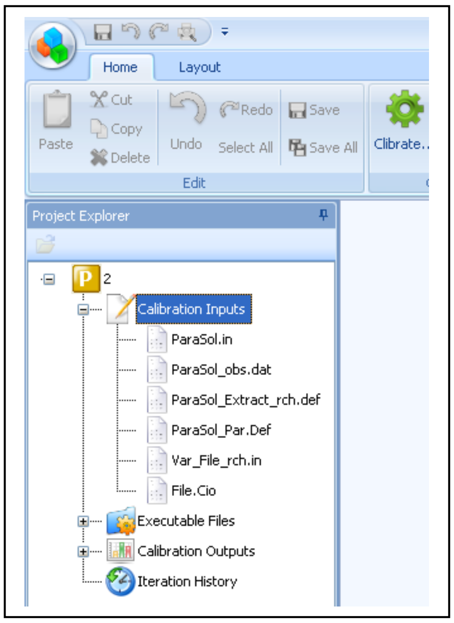
1) Choose ParaSol program type.



2) Edit the input files in "Calibration Files"

3) Execute ParaSol2.exe under "Calibrate"

4) Examine the output in "Calibration Outputs". ParaSol also requires a large number of runs (>5000)



Outputs of ParaSol are in the following files in Para_Sol.OUT:

95ppu.out	Contains the 95% prediction uncertainty of good parameter
ParaSol.out	Detailed outputs
Bestpar.out	File with the best parameter set
Scepar.out	File with all parameter sets used in SCE-UA optimization
Sceobjf.out	File with all objective functions calculated during the SCE-UA optimization
Scegoc.out	File with all objective functions (standardized) and the global optimization criterion (GOC) calculated during the SCE-UA optimization
goodpar.out	File with "good" parameters according to ParaSol
scepargoc.out	File with all parameters and GOC values during SCE runs

summary_stat.out	Summary statistics of all variables

In Figure 17, the execution order and each input and output file of GLUE is listed. The entries are self explanatory.

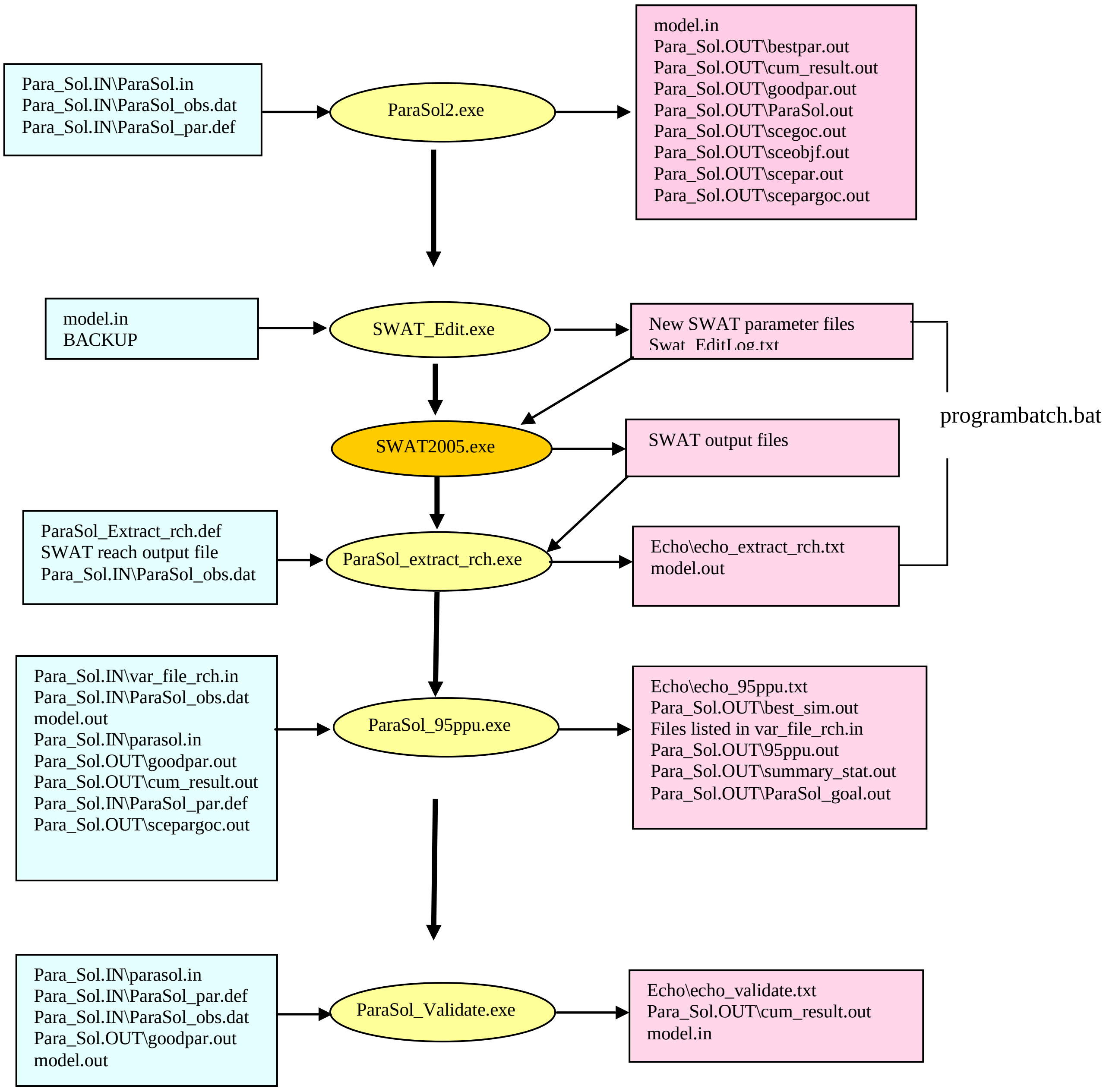


Figure 17. Sequence of program execution and input/output files in ParaSol-SWAT-CUP

VALIDATION

After calibration, validation can be performed by using the “Validate” option from the menu. Before executing validation, however, the ParaSol_obs.dat file must be edited to contain validation data, ParaSol_Extract_rch.def must be edited to extract validation data, and SWAT’s File.cio and climate files (pcp.pcp etc.) must cover the validation period. The validation program uses the good parameters only to run SWAT.

Input files of ParaSol are described below. They are for most parts self explanatory. Read more details about the procedure in the Appendix below.

File Definition

ParaSol.in

10	! Number of parameter to be optimized
1000	! MAXN max no. of trials allowed before optimization is terminated
3	! KSTOP maximum number of shuffling loops in which the criterion value
0.01	! PCENTO percentage by which the criterion value must change...
10	! NGS number of complexes in the initial population
1677	! ISEED initial random seed
5	! NPG number of points in each complex
3	! IPS number of points in a sub-complex
5	!NSPLnumber of evolution steps allowed for each complex before comp
1	! ISTEP1=run optimization + parameter uncertainty 2=rerun model with good parameter sets (see chapter 9)
1	! ISTAT Statistical method for ParaSol (1=Chi-squared; 2=Bayesian)

3	!IPROB iprob, when iprob=1 90% probability ; iprob=2 95% probability;
	iprob=3 97.5% probability , iprob=4 99% probability; iprob=5 99.9 % probability

0	IFLAG Flag indicating whether the objective functions are to be calculated by ParaSol or read from “modelof.out” i=0: objective functions are calculated by LH-OAT based on model.out and data.obs files i>0: i objective functions are read by LH-OAT in the modelof.out file

APPENDIX

ParaSol: optimization and uncertainty analysis tool

Ann van Griensven and Tom Meixner

Department of Environmental Sciences
University of California, Riverside
Riverside, CA92521, USA

Phone: 1-909-787-2356
E-mails: ann@biomath.ugent.be
tmeixner@hwr.arizona.edu

ParaSol files:

File	**description**
ParaSol.exe	Executable for windows
ParaSol.f	Fortran codes for ParaSol.exe
ParaSol.in	Input file for ParaSol.exe
Simple_model.exe	Executable for example model in windows
Simple_model.f	Fortran codes for Simple_model.exe
Batchprogram.bat	Batch file that call simple_model.exe
Input4.	Rainfall inputs for simple_model.exe
Model.in	Input file for simple_model.exe (EAWAG protocol)
Model.out	Output file of simple_model.exe (EAWAG protocol)

Copyrights and terms of use

The users of the programs contained in this diskette can copy and use these programs freely, without seeking authors' permission.
The authors request all users make appropriate references to the use of these programs.
The authors disclaim any responsibility resulting from use of these programs.

Introduction

PS-SG is a tool that performs an optimization and uncertainty analysis for model outputs. In incorporates two methods: ParaSol (Parameter Solutions) that allows for the optimization of model parameters based on SCE-UA algorithm (Duan et al., 1992) and uses the simulations to assess confidence ranges on parameters and outputs (van Griensven and Meixner, 2003a).

Description of the ParaSol method

(van Griensven and Meixner, 2003a)

The ParaSol method aggregates objective functions (OF's) into a global optimization criterion (GOC), minimizes these OF's or a GOC using the SCE-UA algorithm and performs uncertainty analysis with a choice between 2 statistical concepts.

The Shuffled complex evolution (SCE) algorithm

The SCE algorithm is a global search algorithm for the minimization of a single function for up to 16 parameters [*Duan et al*., 1992]. It combines the direct search method of the simplex procedure with the concept of a controlled random search of Nelder and Mead [1965], a systematic evolution of points in the direction of global improvement, competitive evolution [Holland, 1975] and the concept of complex shuffling. In a first step (zero-loop), SCE-UA selects an initial 'population' by random sampling throughout the feasible parameters space for p parameters to be optimized (delineated by given parameter ranges). The population is portioned into several "complexes" that consist of 2p+1 points. Each complex evolves independently using the simplex algorithm. The complexes are periodically shuffled to form new complexes in order to share information between the complexes.

SCE-UA has been widely used in watershed model calibration and other areas of hydrology such as soil erosion, subsurface hydrology, remote sensing and land surface modeling (Duan, 2003). It was generally found to be robust, effective and efficient (Duan, 2003). The SCE-UA has also been applied with success on SWAT for the hydrologic parameters (Eckardt and Arnold, 2001) and hydrologic and water quality parameters (*van* Griensven and Bauwens, 2003).

Objective functions to be used

Within an optimization algorithm it is necessary to select a function that must be minimized or optimized that replaces the expert perception of curve-fitting during the manual calibration. There are a wide array of possible error functions to choose from and many reasons to pick one versus another (for some discussions on this topic see [Legates and McCabe, 1999; Gupta et al., 1998]). The types of objective functions selected for ParaSol are limited to the following due to the statistical assumptions made in determining the error bounds in ParaSol.

Sum of the squares of the residuals (SSQ): similar to the Mean Square Error method (MSE) it aims at matching a simulated series to a measured time series.

$$SSQ = \sum_{i=1,n}\left[x_{i,measured} - x_{i,simulated}\right]^2 \qquad (1)$$

with n the number of pairs of measured ($x_{measured}$) and simulated ($x_{simulated}$) variables

The sum of the squares of the difference of the measured and simulated values after ranking (SSQR): The SSQR method aims at the fitting of the frequency distributions of the observed and the simulated series.

After independent ranking of the measured and the simulated values, new pairs are formed and the SSQR is calculated as

$$SSQR = \sum_{j=1,n}\left[x_{j,measured} - x_{j,simulated}\right]^2 \qquad (2)$$

where j represents the rank.
As opposed to the SSQ method, the time of occurrence of a given value of the variable is not accounted for in the SSQR method (van Griensven and Bauwens, 2003).

Multi-objective optimization

Since the SCE-UA minimizes a single function, it cannot be applied directly for multi-objective optimization. Although there are several methods available in literature to aggregate objective functions to a global optimization criterion (Madsen, 2003; van Griensven and Bauwens, 2003), they do not foresee further application of uncertainty analysis.

A statistically based aggregation method is found within the Bayesian theory (1763). By assuming that the residuals have a normal distribution N(0, σ^2), the variance is estimated as

$$\sigma^2 = \frac{SSQ_{MIN}}{nobs} \tag{3}$$

with SSQ_{MIN} the sum of the squares at the optimum and n_{obs} the number of observations (Box and Tiao, 1973):. The probability of a residual for a given parameter set depends on a specific time series of data and can then be calculated as:

$$p(\theta \mid y_{t,obs}) = \frac{1}{\sqrt{2\Pi\sigma^2}} \exp\left[-\frac{(y_{t,sim} - y_{t,obs})^2}{2\sigma^2}\right] \tag{4}$$

or

$$p(\theta \mid y_{t,obs}) \propto \exp\left[-\frac{(y_{t,sim} - y_{t,obs})^2}{2\sigma^2}\right] \tag{5}$$

for a time series (1..T) this gives

$$p(\theta \mid Y_{obs}) = \frac{1}{\left(\sqrt{2\Pi\sigma^2}\right)^T} \prod_{t=1}^{T} \exp\left[-\frac{(y_{t,sim} - y_{t,obs})^2}{2\sigma^2}\right] \tag{6}$$

or

$$p(\theta \mid Y_{obs}) \propto \exp\left[-\frac{\sum_{t=1}^{T} (y_{t,sim} - y_{t,obs})^2}{2\sigma^2}\right] \tag{7}$$

For a certain time series Y_{obs} the probability of the parameter set θ p(θ|Y_{obs}) is thus proportional to

$$p(\theta \mid Y_{obs}) \propto \exp\left[-\frac{SSQ_1}{2 * \sigma_1^2}\right] \tag{8}$$

where SSQ_1 are the sum of the squares of the residuals with corresponding variance σ_1 for a certain time series. For 2 objectives, a Bayesian multiplication gives:

$$p(\theta \mid Y_{obs}) = C1 * \exp\left[-\frac{SSQ_1}{2*\sigma_1^2}\right] * \exp\left[-\frac{SSQ_2}{2*\sigma_2^2}\right] \tag{9}$$

Applying equation (3), (9) can be written as:

$$p(\theta \mid Y_{obs}) = C2 * \exp\left[-\frac{SSQ_1 * nobs_1}{SSQ_{1,\min}}\right] * \exp\left[-\frac{SSQ_2 * nobs_2}{SSQ_{2,\min}}\right] \tag{10}$$

In accordance to (10), it is true that:

$$\ln[-p(\theta \mid Y_{obs})] = C3 + \frac{SSQ_2 * nobs_2}{SSQ_{2\min}} + \frac{SSQ_2 * nobs_2}{SSQ_{2,\min}} \tag{11}$$

We can thus optimize or maximize the probability of (11) by minimizing a Global Optimization Criterion (GOC) that is set to the equation:

$$GOC = \frac{SSQ_1 * nobs_1}{SSQ_{1,\min}} + \frac{SSQ_2 * nobs_2}{SSQ_{2,\min}} \tag{12}$$

With equation (11), the probability can be related to the GOC according to:

$$p(\theta \mid Y_{obs}) \propto \exp[-GOC] \tag{13}$$

The sum of the squares of the residuals get thus weights that are equal to the number of observations divided by the minimum. The minima of the individual objective functions (SSQ or SSQR) are however initially not known. After each loop in the SCE-UA optimization, an update is performed for these minima of the objective functions using the newly gathered information within the loop and in consequence, the GOC values are recalculated.

The main advantage of using equation 12 to calculate the GOC is that it allows for a global uncertainty analysis considering all objective functions as described below.

Uncertainty analysis method

The uncertainty analysis divides the simulations that have been performed by the SCE-UA optimization into ‘good’ simulations and ‘not good’ simulations and in this way is similar to the GLUE methodology [*Beven and Binley*, 1992]. The simulations gathered by SCE-UA are very valuable as the algorithm samples over the entire parameter space with a focus of solutions near the optimum/optima. To increase the usefulness of the SCE-UA samples for uncertainty analysis, some adaptations were made to the original SCE-UA algorithm, to prevent being trapped in a localized minimum and to allow for a better exploration of the full parameter range and prevent the algorithm from focusing on a very narrow set of solutions. The most important modifications are:

1. After each loop, the m worst results are replaced by random sampling this change prevents the method from collapsing around a local minimum (where m is equal to the number of complexes). Similarly, Vrugt et al. (2003) solved this problem of collapsing in the minimum by introducing randomness. Here however, the randomness was introduced for the replacement of the best results.
2. When parameter values are under or over the parameter range defined by SCE-UA, they get a value equal to the minimum bound or maximum bound instead of a random sampled value .

The ParaSol Algorithm uses two techniques to divide the sample population of SCE-UA into “good” and “bad” simulations. Both techniques are based on a threshold value for the objective function (or global optimization criterion) to select the ‘good’ simulations by considering all the simulations that give an objective function below this threshold. The threshold value can be defined by χ^2-statistics where the selected simulations correspond to the confidence region (CR) or Bayesian statistics that are able point out the high probability density region (HPD) for the parameters or the model outputs (figure 1).

χ^2-method

For a single objective calibration for the SSQ, the SCE-UA will find a parameter set Θ* consisting of the p free parameters (ө*$_1$, ө*$_2$,… ө*$_p$), that corresponds to the minimum of the sum the square SSQ. According to χ^2 statistics (Bard, 1974), we can define a threshold “c” for “good’ parameter set using equation

$$c = OF(\theta^*) * (1 + \frac{\chi^2{}_{p,0.95}}{n-p}) \qquad (14)$$

whereby the $\chi^2_{p,0.95}$ gets a higher value for more free parameters p .

For multi-objective calibration, the selections are made using the GOC of equation (12) that normalizes the sum of the squares for n, equal to the sum of nobs1 and nobs2, observation. A threshold for the GOC is calculated by:

$$c = GOC(\theta^*) * (1 + \frac{\chi^2_{p,0.95}}{nobs1 + nobs2 - p}) \qquad (15)$$

thus all simulations with GOC < X_{gocmin} + are deemed acceptable

Bayesian method

According to the Bayesian theorem, the probability $p(\theta|Y_{obs})$ of a parameter set θ is proportional to equation (11).

After normalizing the probabilities (to ensure that the integral over the entire parameter space is equal to 1) a cumulative distributions can be made and hence a 95% confidence regions can be defined. As the parameters sets were not sampled randomly but were more densely sampled near the optimum during SCE-UA optimisation, it is necessary to avoid having the densely sampled regions dominate the results. This problem is prevented by determining a weight for each parameter set θ_i by the following calculations:

1. Dividing the p parameter range in m intervals
2. For each interval k of the parameter j, the sampling density nsamp(k,j) is calculated by summing the times that the interval was sampled for a parameter j.

A weight for a parameter set θ_i is than estimated by

1. Determine the interval k (between 1 and m) of the parameter θ_i
2. Consider the number of samples within that interval = nsamp(k,j)
3. The weight is than calculated as

$$W(\theta_i) = \left[\prod_{j=1i}^{p} nsamp(k,j) \right]^{1/p} \qquad (16)$$

The "c" threshold is determined by the following process:

a. Sort parameter sets and GOC values according to decreasing probabilities

b. Multiply probabilities by weights

c. Normalize the weighted probabilities by division using PT with

$$PT = \sum_{i=1}^{T} W(\theta_i) * p(\theta_I \mid Y_{obs}) \qquad (17)$$

d. Sum normalized weighted probabilities starting from rank 1 till the sum gets higher than the cumulative probability limit (95% or 97.5%). The GOC corresponding to or closest to the probability limit defines the "c" threshold.

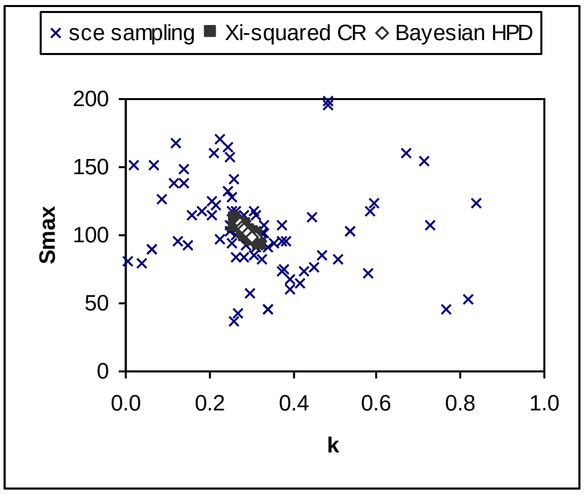


Figure 2: Confidence region CR for the χ^2-statistics and high probability density (HPD) region for the Bayesian statistics for a 2-parameter test model

Using ParaSol

It uses an input file "ParaSol.in". It operates by communicating to the model through input and output files. Input of the model is printed in "model.in" that containes the new parameter values. There are 2 options to communicate with the output:

1. "modelof.out" with the objective functions OR
2. "model.out" with the output values and "data.obs" with the observed values.

For option 2, the model will calculate objective functions based on equation 1.
ParaSol.exe is programmed to run a batchfile "programbatch.bat", containing the necessary commands for the execution of the following:

1. reading the parameters listed in "model.in" and changing the model input files for these parameters values.
2. running the program
3. reading output of the program and printing the objective function(s) into a "modelof.out" file in the right format (if iflag>0)

The ParaSOl package contains an example for the application (simple_model.exe) that is a contains a model with 2 parameters ec [0,200] and ek [0,1], having an optimum at the parameter set (100,0.3). simple_model.exe performs the 3 previously mentioned tasks and is called from the in the “programbatch.bat” file.

For running PS-SG on another applications “otherapplication.exe”, it is thus necessary:

1. To create the appropriate ParaSol.in file, listing all parameters (up to 100) and ranges to be considered and indicating the number of objective functions to consider (up to 40)
2. Having a program “changeinputs.exe” that changes the input files for “otherapplication.exe” according to the values in “ParaSol.in”
3. Having a program “makeobjf.exe” that will read the outputs of “otherapplication.exe”, calculates the objective functions and writes these to the file “modelof.out” (or writes the model.out file with simulations according to the EAWAG format in case of iflag=0).
4. Put the commands “changeinputs.exe”, “otherapplication.exe” and “makeobjf.exe” (if iflag>0) in the “programbatch.dat” file.

Input file formats

Formats for the model.in file (see also example)

Each input (parameter) has one line with parameter name (maximum 40 digits) and the parameter value (free format).

Formats for the modelof.out file (see also example)

1 line with the objective functions in column (free format)

Formats for ParaSol.in

Control parameters

Each control parameter uses one line with free format

MAXN	20000	max no. of trials allowed before optimization is terminated
KSTOP	5	maximum number of shuffling loops in which the criterion value
PCENTO	0.01	percentage by which the criterion value must change...
NGS	10	number of complexes in the initial population
ISEED	1677	initial random seed
NPG	5	number of points in each complex
NPS	8	number of points in a sub-complex
NSPL	5	number of evolution steps allowed for each complex before comp
ISTEP	1	1=run optimization + parameter uncertainty 2=rerun model with good parameter sets (see chapter 9)
ISTAT	1	Statistical method for ParaSol (1=Xi-squared; 2=Bayesian)
IPROB	3	iprob, when iprob=1 90% probability ; iprob=2 95% probability; iprob=3 97.5% probability , iprob=4 99% probability; iprob=5 99.9 % probability
IFLAG	0	Flag indicating whether the objective functions are to be calculated by ParaSol or read from “modelof.out” i=0: objective functions are calculated by LH-OAT based on model.out and data.obs files i>0: i objective functions are read by LH-OAT in the modelof.out file

CHANGEPAR

This section follows the previous section. Each parameter has one row, containing lower limit, upper limit, and the parameter name (up to 250 digits), all in free format.

Output files

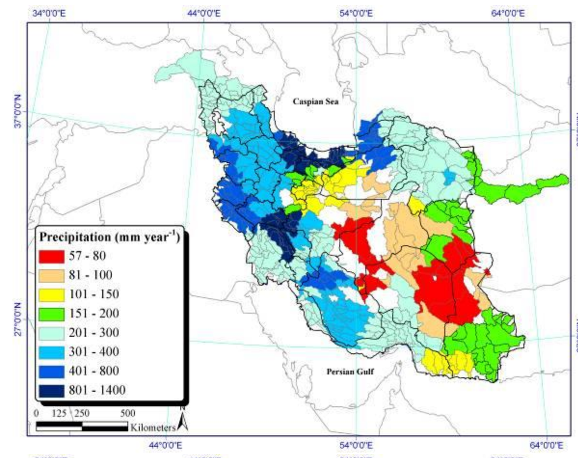
File name	**Description**
ParaSol.out	Detailed outputs.
Bestpar.out	File with the best parameter set
Scepar.out	File with all parameter sets used in SCE-UA optimization
Sceobj.out	File with all objective functions calculated during the SCE-UA optimization
Scegoc.out	File with all objective functions (standardized) and the GOC calculated during the SCE-UA optimization
goodpar.out	File with “good” parameters according to ParaSol
scepargoc.out	File with all parameters and goc values during sce runs.

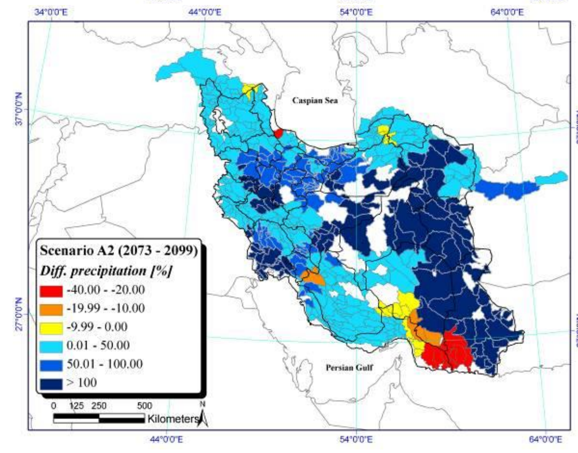
Rerun the model with good parameter sets

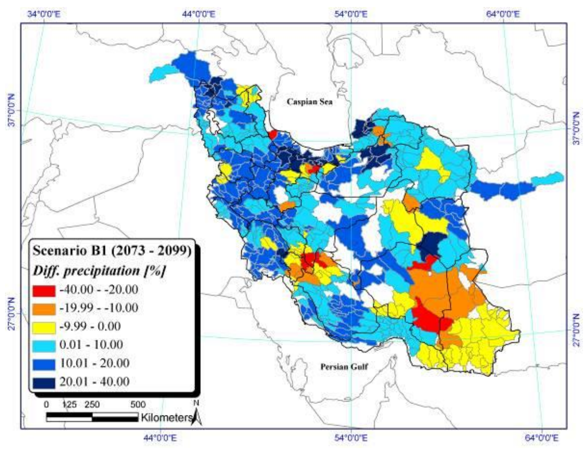
This option only makes sense if you have your model output according to the EAWAG protocol. If you put ISTEP=2 in the ParaSol.in file, the model will rerun all the good parameter sets (in goodpar.out) and calculate the minimum and maximum bounds for the model output (in model.out). These mimimum and maximum values will we printed in the files modelminval.out and modelmaxval.out respectively.

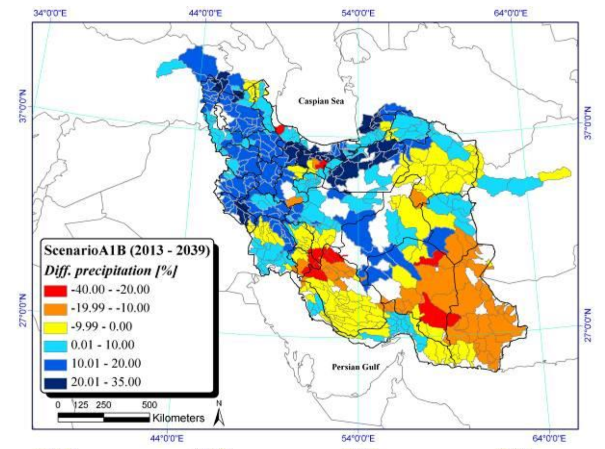
MCMC

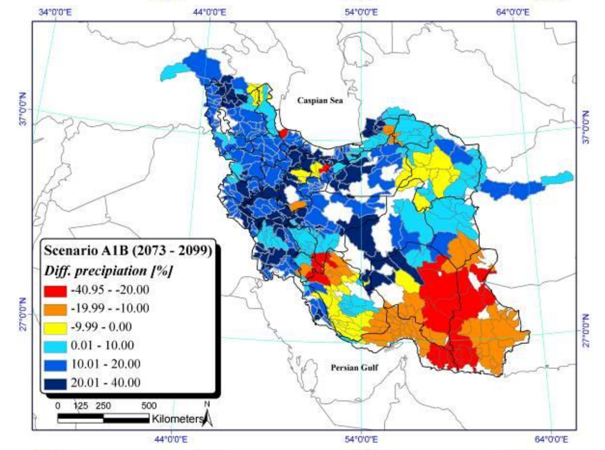
Markov Chain Monte Carlo

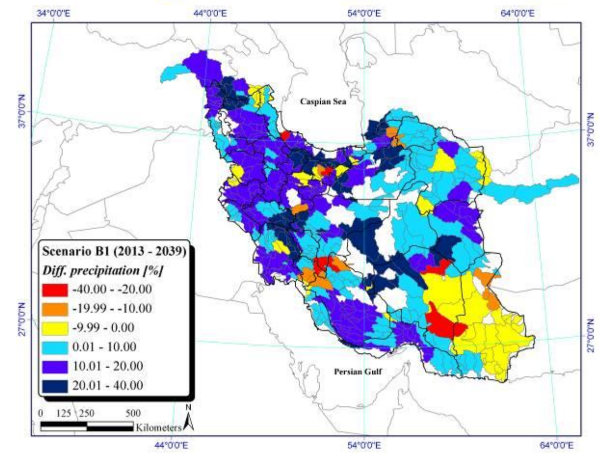







ScenarioA1B (2013 - 2039)
Diff. precipitation [%]
-40.00 - -20.00
-19.99 - -10.00
-9.99 - 0.00
0.01 - 10.00
10.01 - 20.00
20.01 - 35.00
Caspian Sea
Persian Gulf
0 125 250 500 Kilometers
34°0'0"E
44°0'0"E
54°0'0"E
64°0'0"E
37°0'0"N
27°0'0"N


Scenario A1B (2073 - 2099)
Diff. precipitation [%]
-40.95 - -20.00
-19.99 - -10.00
-9.99 - 0.00
0.01 - 10.00
10.01 - 20.00
20.01 - 40.00
Caspian Sea
Persian Gulf
0 125 250 500 Kilometers
34°0'0"E
44°0'0"E
54°0'0"E
64°0'0"E
37°0'0"N
27°0'0"N


Scenario B1 (2013 - 2039)
Diff. precipitation [%]
-40.00 - -20.00
-19.99 - -10.00
-9.99 - 0.00
0.01 - 10.00
10.01 - 20.00
20.01 - 40.00
Caspian Sea
Persian Gulf
0 125 250 500 Kilometers
34°0'0"E
44°0'0"E
54°0'0"E
64°0'0"E
37°0'0"N
27°0'0"N

Introduction to MCMC

MCMC generates samples from a random walk which adapts to the posterior distribution (Kuczera and Parent, 1998). The simplest technique from this class is the Metropolis-Hasting algorithm (Gelman et al. 1995), which is applied in this study. A sequence (Markov Chain) of parameter sets representing the posterior distribution is constructed as follows:

1) An initial starting point in the parameter space is chosen.

2) A candidate for the next point is proposed by adding a random realization from a symmetrical jump distribution, f_{jump}, to the coordinates of the previous point of the sequence:

$$\theta^*_{k+1} = \theta_k + rand(f_{jump}) \tag{13}$$

3) The acceptance of the candidate points depends on the ratio *r*:

$$r = \frac{f_{\mathbf{\Theta}_{\mathrm{post}}|\mathbf{Y}}(\boldsymbol{\theta}^*_{k+1}|\mathbf{y}_{\mathrm{meas}})}{f_{\mathbf{\Theta}_{\mathrm{post}}|\mathbf{Y}}(\boldsymbol{\theta}_k|\mathbf{y}_{\mathrm{meas}})} \tag{14}$$

If *r* >= 1, then the candidate point is accepted as a new point with probability *r*. If the candidate point is rejected, the previous point is used as the next point of the sequence.

In order to avoid long burn-in periods (or even lack of convergence to the posterior distribution) the chain is started at a numerical approximation to the maximum of the posterior distribution calculated with the aid of the shuffled complex global optimization algorithm (Duan et al., 1992).

Step-by-step running of MCMC

The MCMC in SWAT-CUP is based on the procedures developed by Peter Reichert in the UNCSIM package. For more detail we refer the reader to http://www.uncsim.eawag.ch/. To run MCMC the following input files must be created:

mcmc.def

Model	External	
External_ModelInFile	mcmc.in	//parameter file generated internally
External_ModelOutFile	mcmc.out	//simulation file created internally
External_ModelExecFile	mcmc_run.bat	//batch file to start mcmc
ParDefFile	mcmc_par.def	//paerrameter definition file to be prepared by user
PriorDistFile	mcmc_prior.def	//parameter priors to be prepared by user
LikeliDefFile	mcmc_obs.dat	//observation file to be prepared by user
JumpDistFile	mcmc_jump.def	//jump distribution file to be prepared by user
SampSize	100	//number of run to be made by mcmc
ResValFile	mcmc_best.out	//best solution
ResidValFile	mcmc_resid.out	//residual of best solution
PostMarkovChainParSampFile	mcmc_parsamp.out	//Markov Chain of parameters
PostMarkovChainParQuantFile	mcmc_parquant.out	/quantiles of parameter distribution
PostMarkovChainResSampFile	mcmc_ressamp.out	//Markov Chain of result
PostMarkovChainResQuantFile	mcmc_resquant.out	//quantile of Markov Chain residuals
PostMarkovChainPdfSampFile	mcmc_pdfsamp.out	//Markov Chain of pdf of posterior

mcmc_par.def

Name	Value	Minimum	Maximum	Scale	UncRange	Increment	ActSens	ActEstim	Unit	Description
r__CN2.mgt	-0.37213	-0.8	0.2	0.3	0.03	0.03	T	T	0.2	
r__ALPHA_BF.gw	-0.32866	-0.85	0.2	0.325	0.0325	0.0325	T	T	0.2	
r__GW_DELAY.gw	0.404144	-0.2	0.9	0.35	0.035	0.035	T	T	0.9	
r__CH_N2.rte	-0.14402	-0.8	0.8	1	0.1	0.1	T	T	0.8	
v__CH_K2.rte	6.205686	1	10	5.5	0.55	0.55	T	T	10	
……..	……..	……..	……..	……..	……..	……..	……..	……..	……..	
……..	……..	……..	……..	……..	……..	……..	……..	……..	……..	
Lamda1	0.5	0	1	1	0.1	0.1	F	F		
Lamda2	0	0	10	1	0.1	0.1	F	F		
Std_Dev_Out	1	0.1	10	1	0.1	0.1	F	F		

Value - initial estimate of parameter value
Minimum - minimum parameter value
Maximum - maximum parameter value
Scale -
UncRange -
Increment - parameter increment for step changes in Value within Mimimum-Maximum
ActSens -
ActEstim -
Unit -
Description -

mcmc_obs.dat

ResCode	Dat	Transformation	Par_1	Par_2	Dist	Mean	Std_Dev
1	21.41	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
2	23.943	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
3	99.956	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
4	100.169	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
5	53.057	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
6	32.07	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
7	9.286	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
8	1.784	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
9	6.586	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
10	11.948	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
11	14.812	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
12	14.681	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out
……	16.261	BoxCox	Lamda1	Lamda2	Normal	0	Std_Dev_Out

ResCode - label of measured data points
Dat - data value
Transformation - transformation to be performed on the data, i.e., Box Cox transformation
Par_1 - the first parameter of the transformation
Par_2 - the second parameter of the transformation
Dist - distribution of the data point
Mean - mean of the distribution of the data point
Std_Dev - standard deviation of the distribution of the data pint

mcmc_prior.def

Name	Dist	Par_1	Par_2
r__CN2.mgt	Uniform	-0.8	0.2
r__ALPHA_BF.gw	Uniform	-0.85	0.2
r__GW_DELAY.gw	Uniform	-0.2	0.9
r__CH_N2.rte	Uniform	-0.8	0.8
v__CH_K2.rte	Uniform	1	10
r__SOL_AWC.sol	Uniform	-0.2	0.6
………	………	………	………
………	………	………	………

Dist - parameter distribution
Par_1 - first moment of the distribution
Par_2 - second moment of the distribution

Prepare the mcmc_jump.def file according to the following format. A short run maybe necessary first, in order to generate reasonable numbers.

mcmc_jump.def

Name	Dist	Par_1	Par_2
r__CN2.mgt	Normal	0	0.003
r__ALPHA_BF.gw	Normal	0	0.00325
r__GW_DELAY.gw	Normal	0	0.0035
r__CH_N2.rte	Normal	0	0.01
v__CH_K2.rte	Normal	0	0.055
r__SOL_AWC.sol	Normal	0	0.002

Name - parameter name
Dist - parameter distribution
Par_1 - first moment of the distribution
Par_2 - second moment of distribution

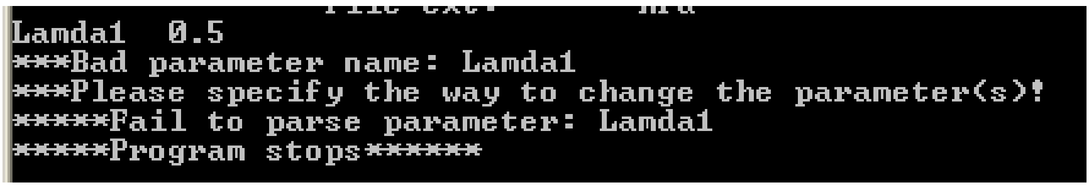
The jump distributions are quite important to convergence and require some initial trial and error runs to specify.

mcmc_run.bat

SWAT_Edit.exe
swat2005.exe
MCMC_extract_rch.exe
	//program to insert generated parameters in swat input files
	//swat program either swat2000 or swat2005
	//program to extract the desired outputs from swat output files

7- Run the program executing mcmc_start.bat

Note: Please ignore the following error during the run:



```
Lamda1  0.5
***Bad parameter name: Lamda1
***Please specify the way to change the parameter(s)!
*****Fail to parse parameter: Lamda1
*****Program stops*******
```

In Figure 18, the execution order and each input and output file of GLUE is listed. The entries are self explanatory.

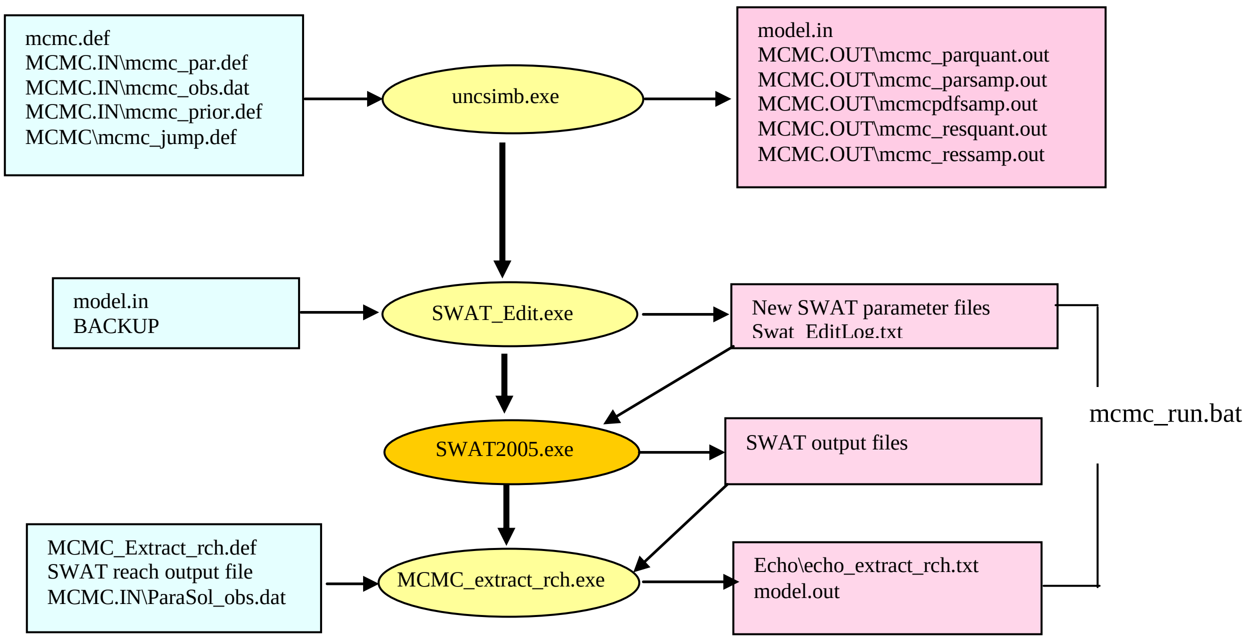


Figure 18. Sequence of program execution and input/output files in MCMC-SWAT-CUP

Reference

Abbaspour, K.C., J. Yang, I. Maximov,., R. Siber, K. Bogner, J. Mieleitner, J. Zobrist, R. Srinivasan. 2007. Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT. *Journal of Hydrology*, 333:413-430.

Abbaspour, K.C., 2005. Calibration of hydrologic models: when is a model calibrated? In Zerger, A. and Argent, R.M. (eds) MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, December 2005, pp. 2449-12455. ISBN: 0-9758400-2-9. http://www.mssanz.org.au/modsim05/papers/abbaspour.pdf

Abbaspour, K.C., Johnson, A., van Genuchten, M.Th, 2004. Estimating uncertain flow and transport parameters using a sequential uncertainty fitting procedure. Vadose Zone Journal 3(4), 1340-1352.

Abbaspour, K. C., R. Schulin, M. Th. Van Genuchten, 2001. Estimation of unsaturated soil hydraulic parameters using ant colony optimization. Advances in Water Resources, 24: 827-841.

Abbaspour, K. C., M. Sonnleitner, and R. Schulin. 1999. Uncertainty in Estimation of Soil Hydraulic Parameters by Inverse Modeling: Example Lysimeter Experiments. *Soil Sci. Soc. of Am. J.*, 63: 501-509.

Abbaspour, K. C., M. Th. van Genuchten, R. Schulin, and E. Schläppi. 1997. A sequential uncertainty domain inverse procedure for estimating subsurface flow and transport parameters. *Water Resour. Res*., v. 33, no. 8., pp. 1879-1892.

Arnold, J.G., Srinivasan R., Muttiah R.S., Williams J.R., 1998. Large area hydrologic modeling and assessment - Part 1: Model development. Journal of the American Water Resources Association 34(1), 73-89.

Bard, 1974. *Non Linear Parameter Estimation*. Academic Press, New York N.Y.

Box, G.E.P., and G.C.Tiao. *Bayesian Inference in Statistical Analysis*, Addison-Wesley-Longman, Reading, Mass, 1973.

Beven, K. and Freer, J., 2001. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the GLUE methodology. Journal of Hydrology, 249(1-4): 11-29.

Beven, K. and Binley, A., 1992. The Future of Distributed Models - Model Calibration and Uncertainty Prediction. Hydrological Processes, 6(3): 279-298.

Duan, Q., Global Optimization for Watershed Model Calibration, in *Calibration of Watershed Models*, edited by Q. Duan, H. V. Gupta, S. Sorooshian, A. N. Rousseau, and R. Turcotte, pp. 89-104, AGU, Washington, DC, 2003.

Duan, Q., V. K. Gupta, and S. Sorooshian, Effective and efficient global optimization for conceptual rainfall-runoff models, *Water. Resourc. Res., 28*:1015-1031, 1992.

Duan, Q., S. Sorooshian, H. V. Gupta, A. N. Rousseau, and R. Turcotte, *Advances in Calibration of Watershed Models,*AGU, Washington, DC, 2003.

Eckhardt K and J.G. Arnold. Automatic calibration of a distributed catchment model. , *J. Hydrol.*, 251: 103-109. 2001.

Faramarzi, M., K.C. Abbaspour, H. Yang, R. Schulin. 2008. Application of SWAT to quantify internal renewable water resources in Iran. Hydrological Sciences. DOI: 10.1002/hyp.7160.

Gelman, S., Carlin, J.B., Stren, H.S., Rubin, D.B., 1995. Bayesian Data Analysis, Chapman and Hall, New York, USA.

Gupta, H. V., S. Sorooshian, and P. O. Yapo, Toward improved calibration of hydrologic models: multiple and noncommensurable measures of information, *Water. Resourc. Res., 34*:751-763, 1998.

Holland, J.H. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 183 p, 975, 1975.

Hornberger, G.M. and Spear, R.C., 1981. An Approach to the Preliminary-Analysis of Environmental Systems. Journal of Environmental Management, 12(1): 7-18.

Kuczera, G., Parent, E., 1998. Monte Carlo assessment of parameter uncertainty in conceptual catchment models: the Metropolis algorithm. Journal of Hydrology, 211(1-4): 69-85.

Legates, D. R. and G. J. McCabe, Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation, *Water. Resourc. Res., 35*:233-241, 1999.

Madsen, H., Parameter estimation in distributed hydrological catchment modelling using automatic calibration with multiple objectives. Advances in water resources, 26, 205-216, 2003.

Marshall, L., D. Nott, and A. Sharma 2004. A comparative study of Markov chain Monte Carlo methods for conceptual rainfall-runoff modeling. Water Resources Research, 40, W02501, doi:10.1029/2003WR002378.

McKay, M.D., Beckman, R. J., Conover, W.J., 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics. 21, 239-245.

Nash, J. E., J. V. Sutcliffe, 1970. River Flow Forecasting through Conceptual Models 1. A Discussion of Principles. Journal of Hydrology 10(3), 282-290.

Nelder, J.A., R. A. Mead, simplex method for function minimization, *Computer Journal*, 7, 308-313, 1965.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., 1992. Numerical Recipe, The Art of Scientific Computation. 2nd ed. Cambridge University Press, Cambridge, Great Britain.

Romanowicz, R. J., Beven K., and Tawn J. 1994. Evaluation of Predictive Uncertainty in Nonlinear Hydrological Models Using a Bayesian Approach. In: *Statistics for the Environment 2, Water Related Issues* ,eds V. Barnett and K. F. Turkman, 297-315, Wiley, Chichester.

Rouholahnejad, E., K.C. Abbaspour, M. Vejdani, R. Srinivasan, R. Schulin, and A. Lehmann. 2011. Parallelizing SWAT calibration in Windows using the SUFI2 program. Environmental Modelling and Software. Submitted.

Schuol, J., K.C. Abbaspour, R. Srinivasan, and H.Yang. 2008a. Modelling Blue and Green Water Availability in Africa at monthly intervals and subbasin level. Water Resources Research. VOL. 44, W07406, doi:10.1029/2007WR006609.

Schuol, J., Abbaspour, KC., Sarinivasan, R., Yang, H. 2008b. Estimation of freshwater availability in the West African Sub-continent using the SWAT hydrologic model. *Journal of Hydroloy*. 352(1-2):30-49.

van Griensven A. and W. Bauwens. 2003. Multi-objective auto-calibration for semi-distributed water quality models, *Water. Resourc. Res*. 39 (12): Art. No. 1348 DEC 16.

Van Griensven, A., Meixner, T., 2006. Methods to quantify and identify the sources of uncertainty for river basin water quality models. Water Science and Technology, 53(1): 51-59.

Vrugt, J. A., H. V. Gupta, W. Bouten, and S. Sorooshian. 2003. A shuffled Complex Evolution Metropolis Algorithm for Estimating Posterior Distribution of Watershed Model Parameters, in *Calibration of Watershed Models* , ed. Q. Duan, S. Sorooshian, H. V. Gupta, A. N. Rousseau, and R. Turcotte, AGU Washington DC, DOI: 10.1029/006WS07.

Yang, J., Reichert, P., Abbaspour, K.C., Yang, H., 2007. Hydrological Modelling of the Chaohe Basin in China: Statistical Model Formulation and Bayesian Inference. Journal of Hydrology, 340: 167-182.

Yang, J., Abbaspour K. C., Reichert P., and Yang H. 2008. Comparing uncertainty analysis techniques for a SWAT application to Chaohe Basin in China. In review. Journal of Hydrology. 358(1-2):1-23.

Yapo, P. O., Gupta, H.V., Sorooshian, S., 1998. Multi-objective global optimization for hydrologic models. J. of Hydrol. 204, 83-97.