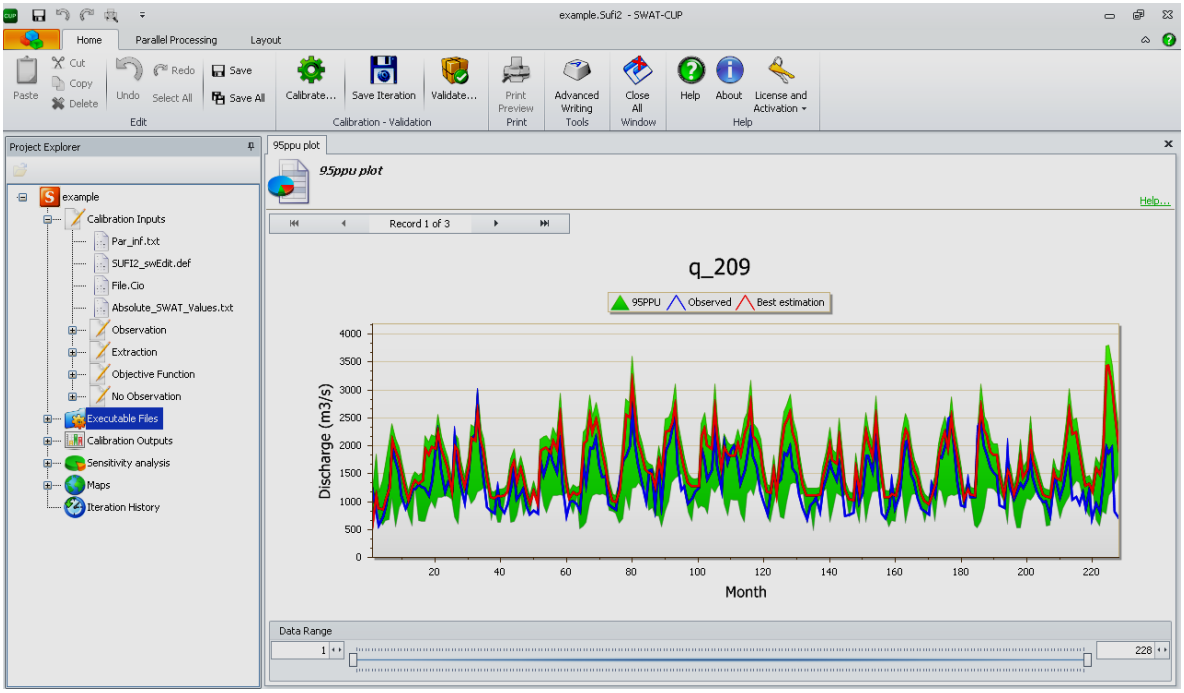
SWAT-CUP

SWAT Calibration and Uncertainty Programs



Karim C. Abbaspour

abbaspour_kc@gmail.com

DISCLAIMER

This report documents SWAT-CUP, a computer program for calibration of SWAT models. 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-CUP 2012 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 write to SWAT-CUP Google group at swat-cup@googlegroups.com

To purchase Parallel Processing or to download the newest version of the program please visit the website www.2w2e.com

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.

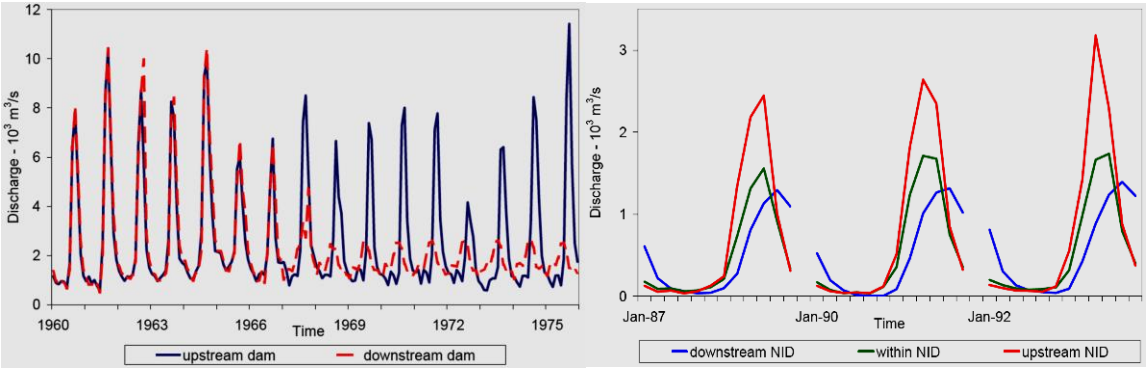


Figure 1. left) Effect of Aswan dam on down stream discharge before and after its operation in 1967. right) The influence of Niger Inland Delta on the through flow at upstream, within, and downstream of the wetland. (After Schuol et al., 2008a,b)

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.

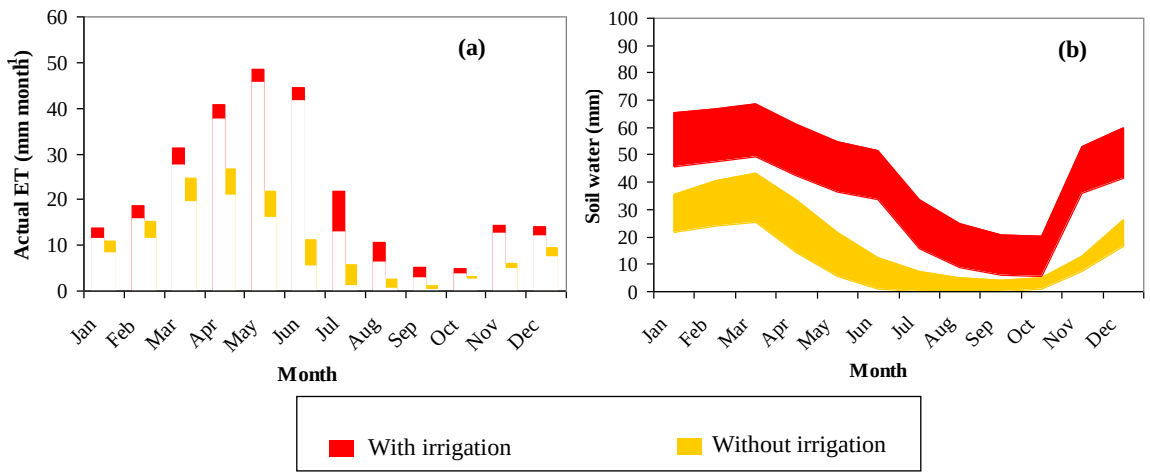


Figure 2. Illustration of the differences in predicted actual ET (a) and soil moisture (b) with and without considering irrigation in Esfahan province, Iran. The variables are monthly averages for the period of 1990-2002. (After Faramarzi et al., 2009)

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.

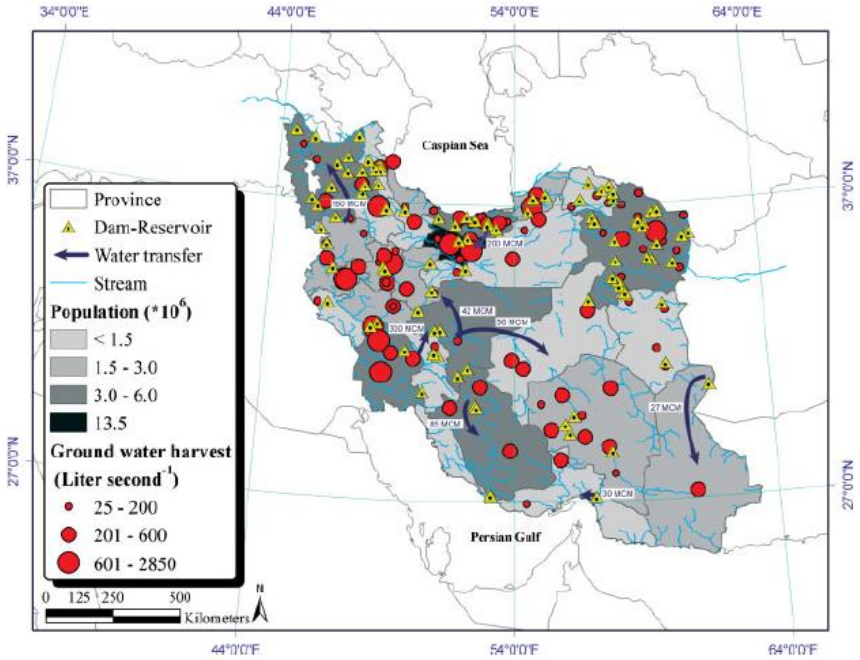


Figure 3. Water management map of Iran showing some of man's activities during 1990-2002. The map shows locations of dams, reservoir, water transfers and groundwater harvest (background shows provincial-based population). After Faramarzi et al., (2009).

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.

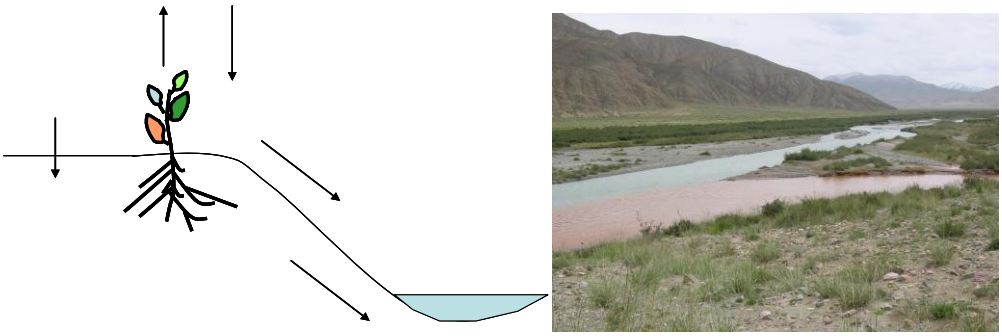


Fig. 4. (left) A simplified conceptual model of hydrology in a watershed where revap is ignored. (right) A natural process near the source of Yellow River in China playing havoc with river loading based on the USLE!

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

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

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

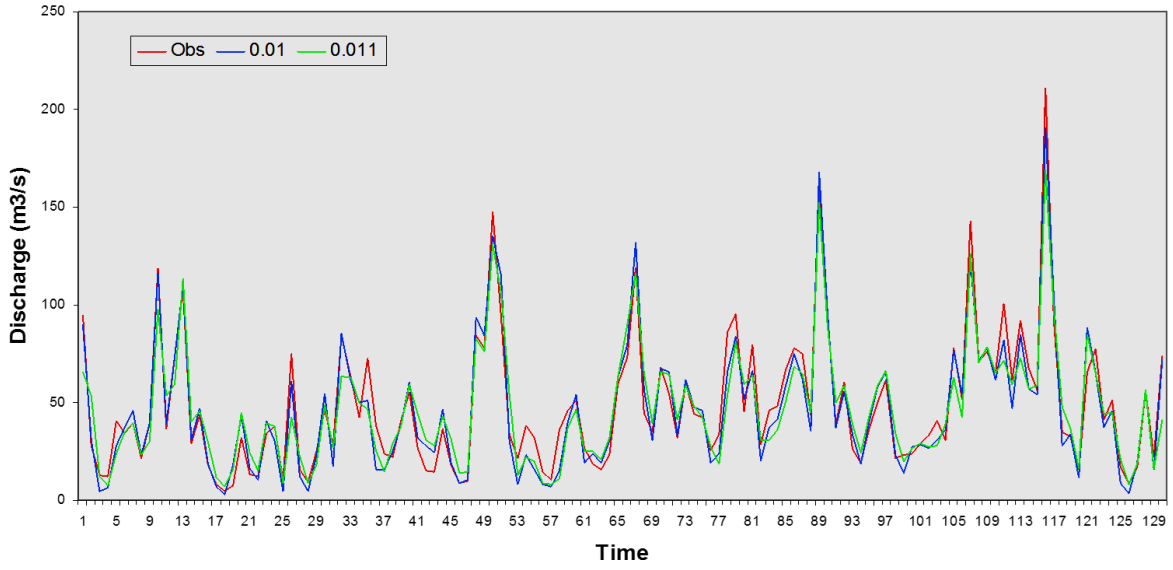


Figure 9. Example of parameter non-uniqueness showing two similar discharge signals based on quite different parameter values

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.

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.

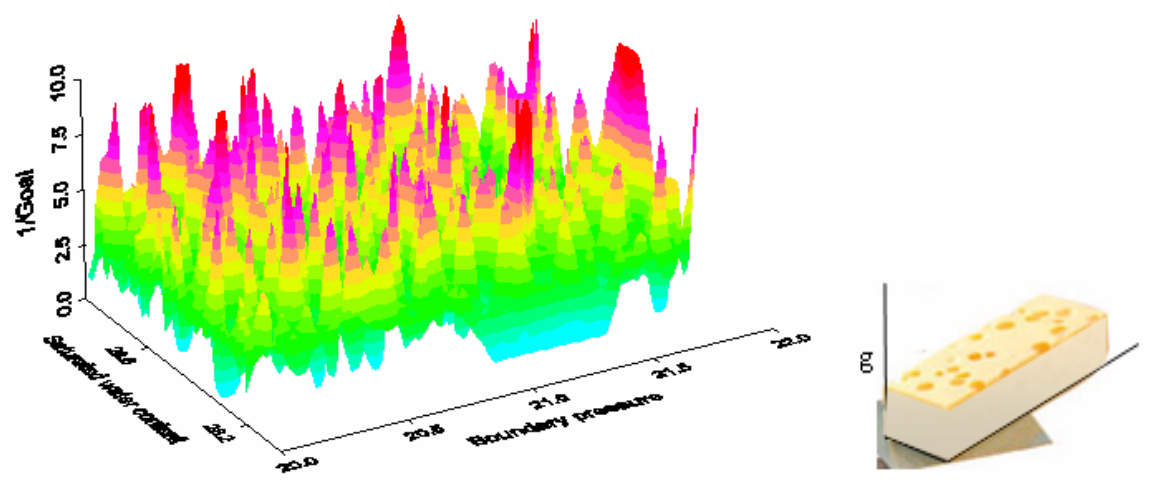


Figure 10. A multidimensional objective function is "multimodal" meaning that there are many areas of good solutions with different uncertainties much like the mysterious holes in a slice of Swiss cheese.

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	-0.59 (-0.60, -0.58)	-0.59 (-0.60, -0.58)	-0.52 (-0.60, 0.03)	-0.55 (-0.56, 0.15)	0.58
a__CH_K2.rte	-0.45 (-0.56, 0.46)	35.70 (27.72,37.67)	83.95 (69.4, 150.0)	78.3 (68.0, 86.2)	147.23
a__OV_N.hru	78.19 (6.0, 144.8)	0.11 (0.07, 0.10)	0.06 (0.00, 0.11)	0.12 (0.00, 0.19)	0.08
[2]σ_{dry}	0.05 (0.00, 0.20)	-	-	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).

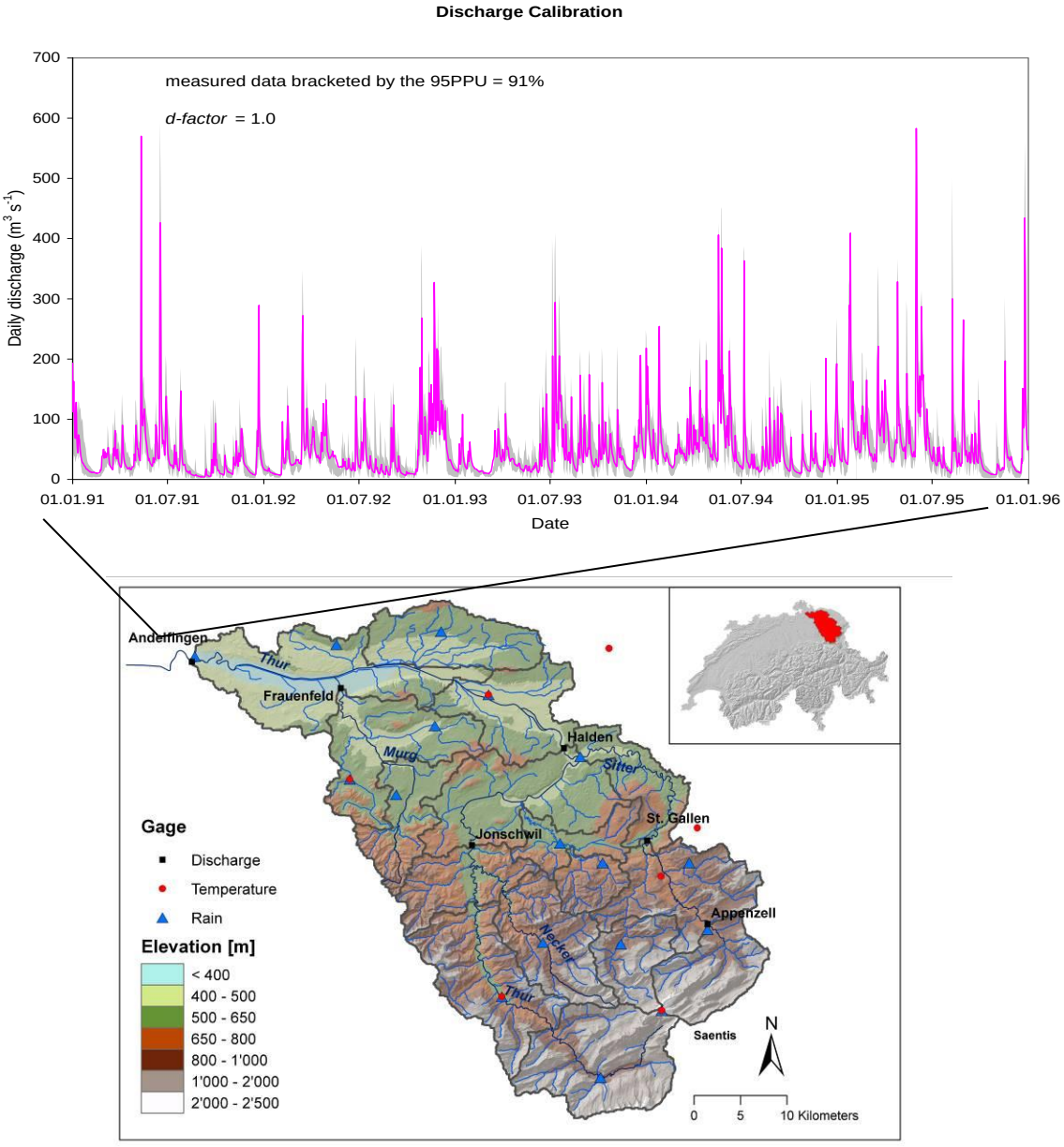
[2] the σ_{dry}, σ_{wet}, τ_{dry}, and τ_{wet} used to calculate the Calculate the logarithm of the posterior probability density function (PDF) are from the best of MCMC.

[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

The "deterministic" approach to calibration is now outdated and unacceptable. Example of a deterministic optimization is "trial and error". Meaning you keep adjusting parameters until you get some kind of a reasonable match between simulation and observation. Reporting this as a calibrated model, in my opinion is wrong, and will not stand in any court of law, if it comes to that. Here, we will not further discuss the deterministic approaches that result in a single set of parameters claiming to represent the "best simulation".

In "stochastic" calibration, we recognize the errors and uncertainties in our modeling work and try to capture, to some degree, our ignorance and lack of understanding of the processes in natural systems. There is an intimate relationship between calibration and uncertainty (Abbaspour, et al., 2015). Reporting the uncertainty is not a luxury in modeling, it is a necessity. Without the uncertainty, calibration is meaningless and misleading. Furthermore, any analysis with the calibrated model must include the uncertainty in the result by propagating the parameter uncertainties.

In SUFI-2, uncertainty in parameters, expressed as ranges (uniform distributions), accounts for all sources of uncertainties such as uncertainty in driving variables (e.g., rainfall), conceptual model, parameters, and measured data. Propagation of the uncertainties in the parameters leads to uncertainties in the model output variables, which are expressed as the 95% probability distributions. These are calculated at the 2.5% and 97.5% levels of the cumulative distribution of an output variable generated by the propagation of the parameter uncertainties using Latin hypercube sampling. This is referred to as the 95% prediction uncertainty, or 95PPU. These 95PPUs are the model outputs in a stochastic calibration approach. It is important to realize that we do not have a single signal representing model output, but rather an envelope of good solutions expressed by the 95PPU, generated by certain parameter ranges.

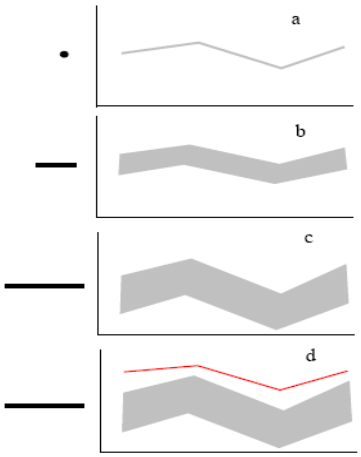
In SUFI2, we want that our model result (95PPU) envelops most of the observations. Observation, is what we have measured in the natural system. Observation is important because it is the culmination of all the processes taking place in the region of study. The argument, however naïve, is that if we capture the observation correctly with our model, then we are somehow capturing correctly all the processes leading to that observation. The problem, of course, is that often a combination of wrong processes in our model may also produce good simulation results. For this reason, the more variables (representing different processes) we include in the objective function, the more likely we are to avoid the wrong processes.

To quantify the fit between simulation result, expressed as 95PPU, and observation expressed as a single signal (with some error associated with it) we came up with two statistics: *P-factor* and *R-factor* (see Abbaspour et al., 2004, 2007 references provided in the reference list of SWAT-CUP). *P-factor* is the percentage of observed data enveloped by our modeling result, the 95PPU. *R-factor* is the thickness of the 95PPU envelop. In SUFI2, we try to get reasonable values of these two factors. While we would like to capture most of our observations in the 95PPU envelop, we would at the same time like to have a small envelop. No hard numbers exist for what these two factors should be, similar to the fact that no hard numbers exist for R^2 or NS. The larger they are, the better they are. For *P-factor*, we suggested a value of >70% for discharge, while having *R-factor* of around 1. For sediment, a smaller *P-factor* and a larger *R-factor* could be acceptable.

SUFI2 operates by performing several iterations, usually at most <5. In each iteration, the parameter ranges get smaller zooming on a region of the parameter space, which produced better results in the

previous iteration. Naturally, as parameter ranges get smaller, the 95PPU envelop gets smaller, leading to smaller *P-factor* and smaller *R-factor*. As each iteration zooms into a better region of the parameter space, obtained by the previous iteration, it is going to find a better "best" solution. So, if you have NS as your objective function, then you will get a better NS in subsequent iterations, but the *P-factor* and *R-factor* will decrease because of narrower parameter ranges. But the idea is not to find that so called "best simulation". Because, 1) there are always better simulations, and 2) the difference between the "best simulation" and the "next best simulation" and the "next next best simulation" is usually statistically insignificant (e.g., NS=0.83 vs NS=0.81 are probably not significantly different), meaning that they could both be identified as the best simulations. But while the differences are insignificant in terms of the objective function value, they are very significant in terms of parameters values. Therefore, the next-best solution cannot be ignored.

The concept behind the uncertainty analysis of the SUFI-2 algorithm is depicted graphically in the Figure below. This Figure illustrates that a single parameter value (shown by a point) leads to a single model response (Fig. a), while propagation of the uncertainty in a parameter (shown by a line) leads to the 95PPU illustrated by the shaded region in Figure b. As parameter uncertainty increases, the output uncertainty also increases (not necessarily linearly) (Fig. c). 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).



A conceptual illustration of the relationship between parameter uncertainty and prediction uncertainty

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*.

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 d occurs, then the problem is not one of parameter calibration and the conceptual model must be re-examined.

SUFI-2 as an optimization algorithm

For a description of SUFI-2 see Abbaspour et al., (2004, 2007). References are provided in the Reference directory of SWAT-CUP.

For a calibration protocol see:
http://www.sciencedirect.com/science/article/pii/S0022169415001985

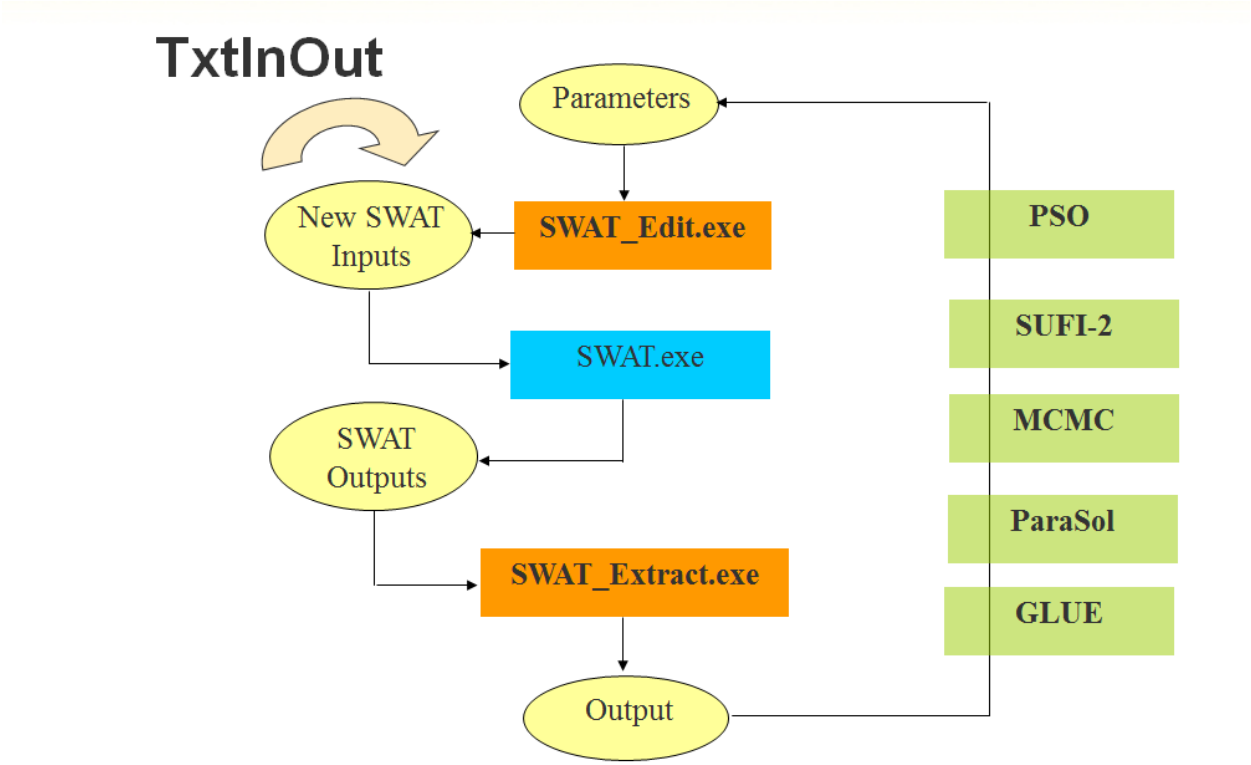
and

https://www.mdpi.com/2073-4441/10/1/6

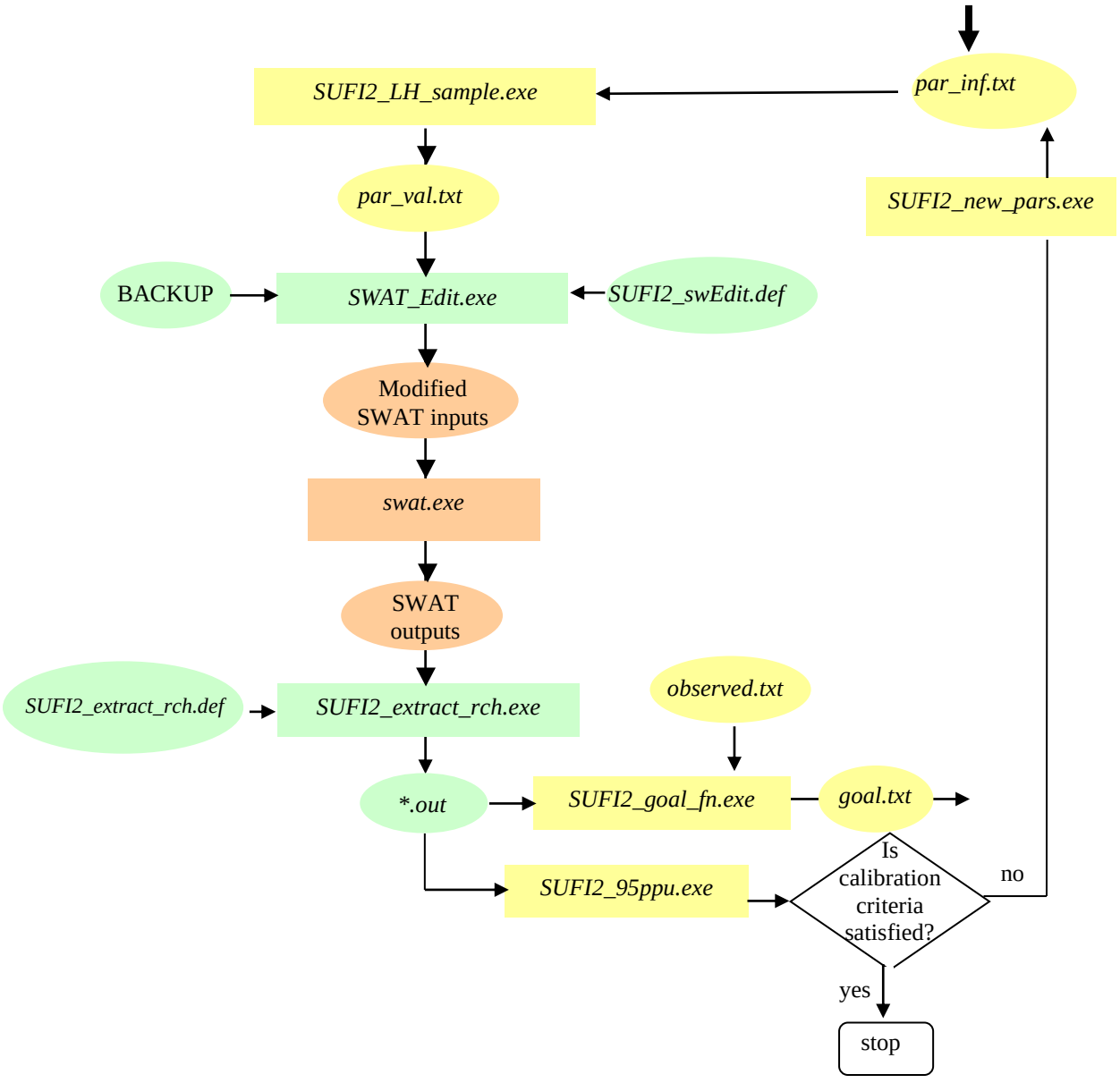
SWAT-CUP

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 five optimization programs is illustrated in the Figure below.



Step-by-step Creating of SWAT-SUFI2 Input Files



1. Before SWAT-CUP

Become familiar with SWAT parameters. They are all explained in the SWAT I/O manual. Also, 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)
https://www.eawag.ch/fileadmin/Domain1/Abteilungen/siam/software/swat/thur.pdf

- The application to entire Europe (Abbaspour et al., 2015)
(http://www.sciencedirect.com/science/article/pii/S0022169415001985)

- A Guideline for Successful Calibration and Uncertainty Analysis for Soil and Water Assessment: A Review of Papers from the 2016 International SWAT Conference
https://www.mdpi.com/2073-4441/10/1/6

- SUFI-2 program, Landfill application in Switzerland (Abbaspour et al., 2004)
https://pubs.geoscienceworld.org/vzj/article-abstract/3/4/1340/111784/estimating-uncertain-flow-and-transport-parameters?redirectedFrom=fulltext

- SWAT applications in Africa (Schuol et al, 2008a,b)
https://www.sciencedirect.com/science/article/pii/S0022169407007755
https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2007WR006609

- The comparison of different optimization programs
https://www.sciencedirect.com/science/article/pii/S0022169408002370?via%3Dihub

- The parallel processing paper
https://www.sciencedirect.com/science/article/pii/S1364815211002829

- The Black Sea application paper
https://agupubs.onlinelibrary.wiley.com/doi/10.1002/2013WR014132

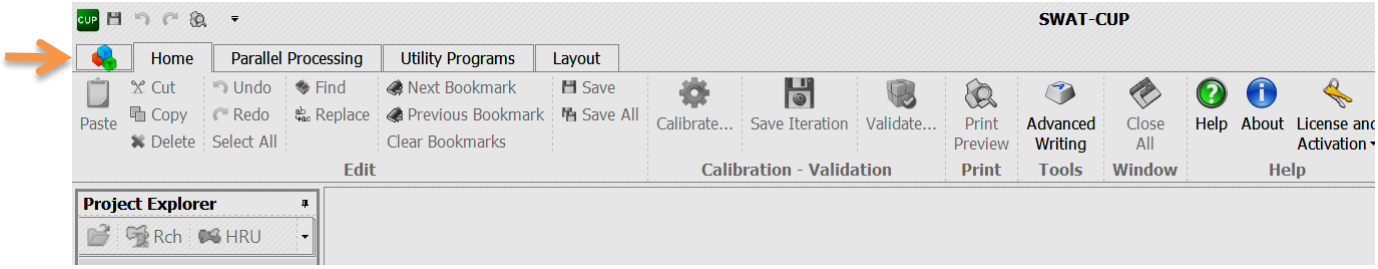
And others provided in the C:\SWAT\SWAT-CUP\ExternalData\References

2. Start SWAT-CUP

Install the SWAT-CUP in C:\SWAT\SWAT-CUP, the same directory as the SWAT and start the program by pressing the SWAT-CUP icon on the desktop:

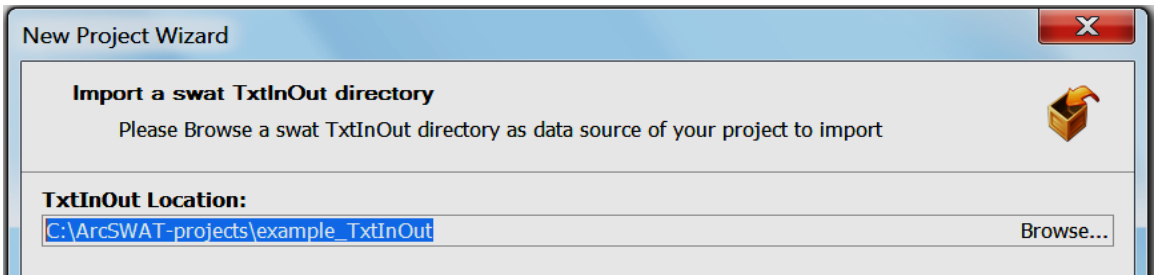
3. Open a Project

- To open a new or old project: Press the  symbol at the top left corner

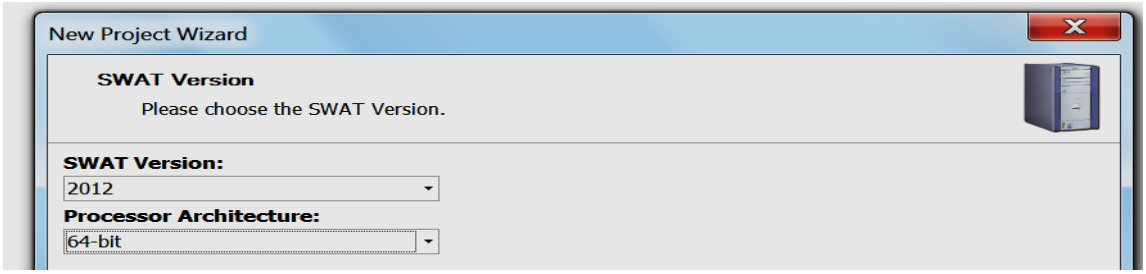


and choose a “New” or “Open” an old project

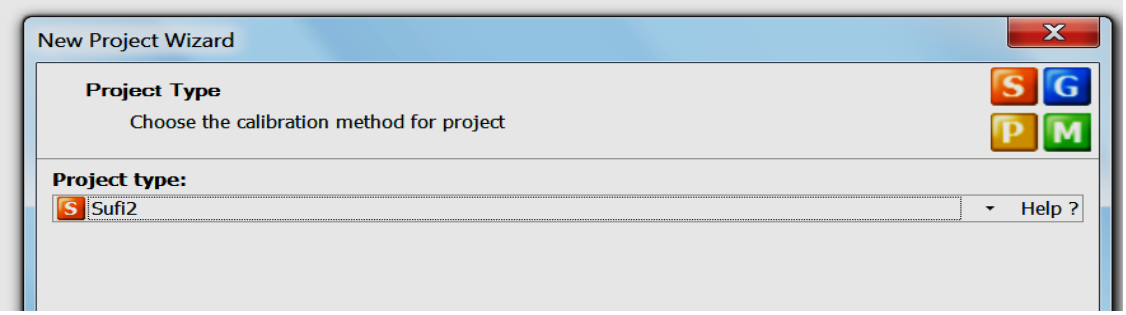
- For a new project locate a SWAT “TxtInOut” directory. Any file with “TxtInOut” in the name string would be acceptable



- Choose SWAT and processor versions

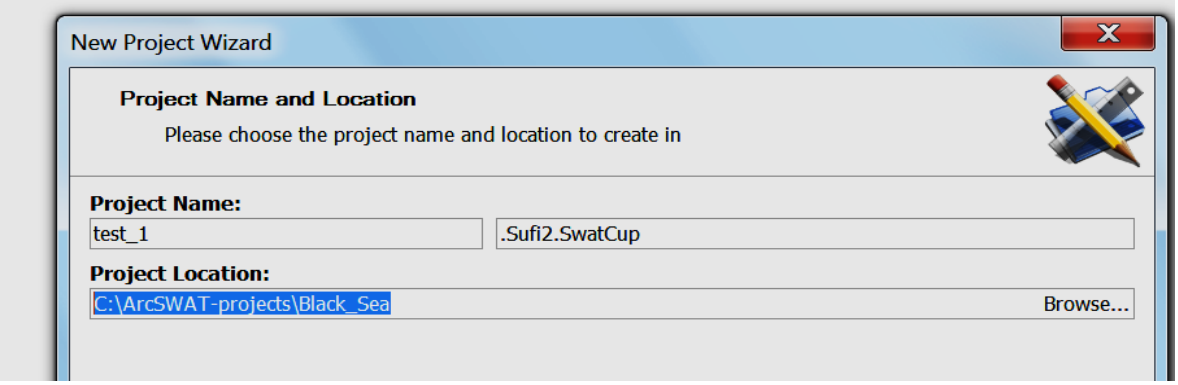


- Select a program from the list provided (SUFI2,GLUE, ParaSol, MCMC, PSO). A detailed explanation of the SUFI-2 procedure is offered here, but all programs follow the same format.



- Give a name to the project and a location where SWAT-CUP project can be saved.

Note the default addition to the name provided in the window to the right of "Project Name" window. For a SUFI2 project, the full SWAT-CUP directory name in the example below would be **test_1.Sufi2.SwatCup**, which will reside in **c:\ArcSWAT-projects\Black_Sea** directory.

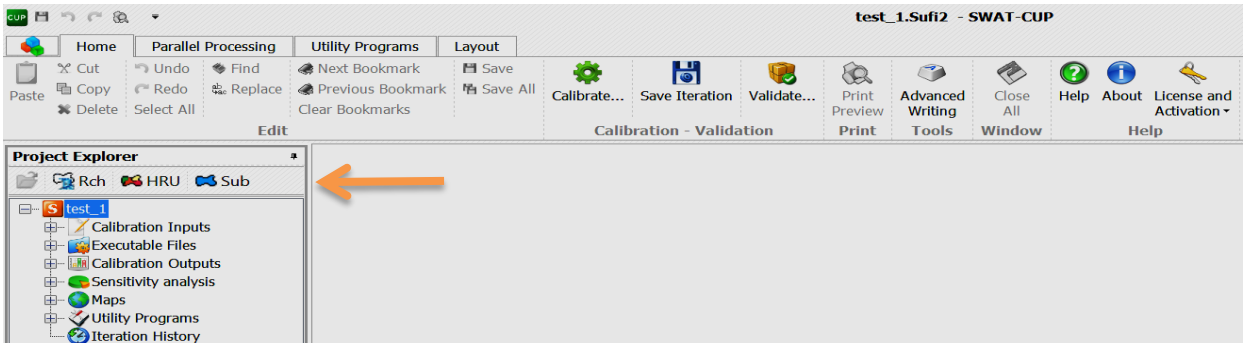


At this point The program creates the desired project directory and copies there all TxtInOut files from the indicated location into the SWAT-CUP project directory. It also creates a directory called "Backup" in the same SWAT-CUP project directory and copies all SWAT TxtInOut file there. The parameters in the files in the Backup directory serve as the default parameters and do not changed during the calibration process. The Backup directory is always needed - as its original form – because, relative changes that have been made to the parameters during calibration, were made relative to the parameter values in the Backup directory. Therefore, it is important that the Backup directory is never changed.

4. SWAT Output Files

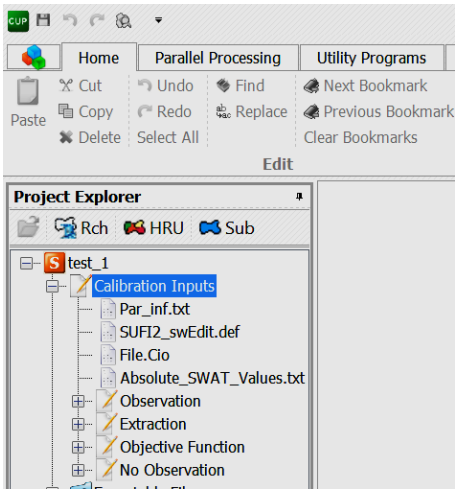
You can calibrate the model based on the variables from output.rch, output.hru, output.sub, output.res, output.mgt, and now also hourly-generated files. However, the interface only shows .rch, .hru, and .sub files. As most often we have either discharge, nutrients data, or sediment data, only these are shown in

the interface. Just click and activate which file you want to use (i.e., your observed variables reside in which SWAT file(s)).



5. Calibration Inputs

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



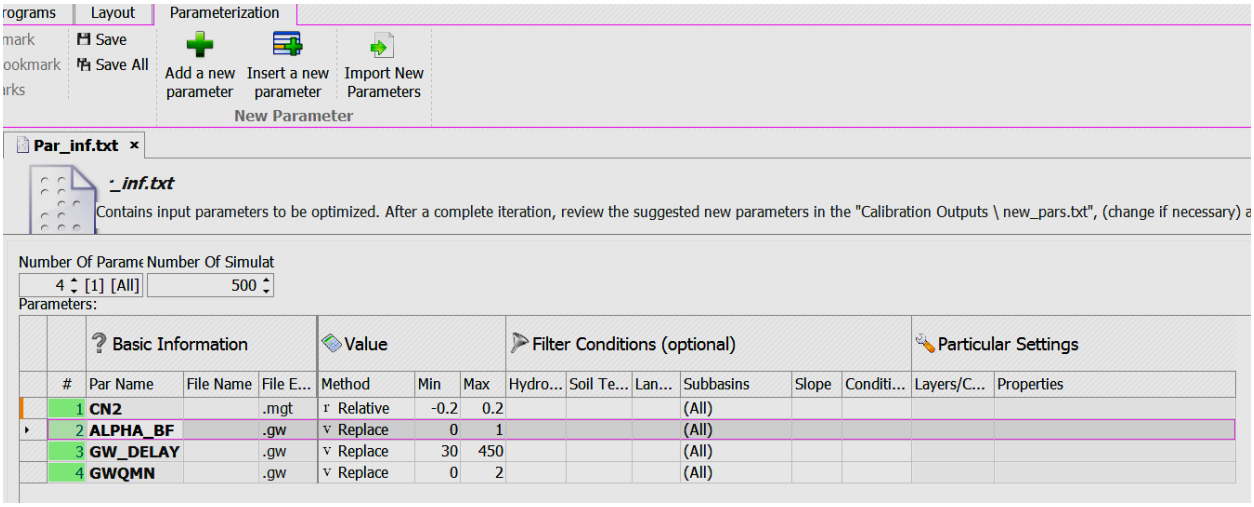
- Par_inf.txt

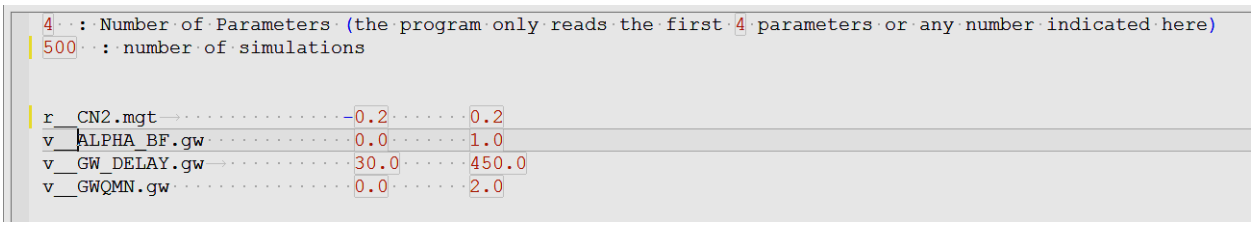
This file resides in the project directory in **SUFI2.IN** directory. It contains input parameters to be optimized. An example is provided, which needs to be edited by the user. The examples shows the format of the file. Edit this to your needs. A *text view* and a *form view* is provided. The *form view* helps with finding the correct parameter syntax or expression. All SWAT parameters (up to the time of compilation of the last swat-cup version) can be found here. The form view follows standard Windows protocol and the users are advised to familiarize themselves with this module by testing different features of it. What you do in the *form view*, appears in the *text view* and vice versa. Users are also encouraged to try different things in the *form view* and look at the them in the *text view* to become familiar with this important and unique feature of SWAT-CUP.

This file contains the number of parameters to be optimized and the number of simulations to make in the current iteration. SUFI2 is iterative, each iteration contains a number of simulations. Around 500

simulations are recommended in each iteration. But if a swat project takes too long to run, fewer number of simulations (200-300) in each iteration could be acceptable. Parameters are sampled using Latin hypercube scheme explained later in the manual. Usually, not more than 4 iterations are sufficient to reach an acceptable solution. A parallel processing module is also available to speed up the calibration process.

To learn more about "parameterization" and parameter qualifiers r__, v__, and a__ please see the section on "parameterization" below.

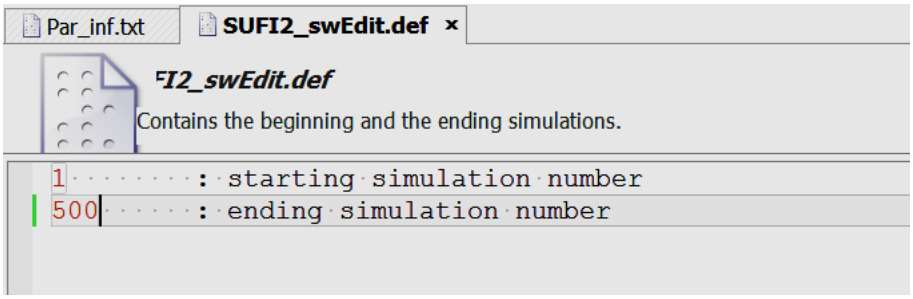




What parameters to use depend on the objective function. Initially, in every case, flow should be calibrated and then water quality variables added one at time (see Abbaspour et al., 2007, 2015 for parameter choices and calibration protocol).

- SUFI2_swEdit.def

This file contains the beginning and the ending simulation years. Please note that **the beginning simulation does not include the warm up period.** SWAT simulates the warm up period but does not print any results, therefore, these years are not considered in SWAT-CUP. You can check output.rch file of SWAT to see when the beginning and the ending simulation times is.



- **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 years (NYSKIP) to correctly provide SWAT-CUP with beginning and end year of simulation. It is recommended that you have 2-3 years of warm up period.

```
General Information/Watershed Configuration:
fig.fig
              15     | NBYR : Number of years simulated
            1987     | IYR : Beginning year of simulation
               1     | IDAF : Beginning julian day of simulation
             365     | IDAL : Ending julian day of simulation
```

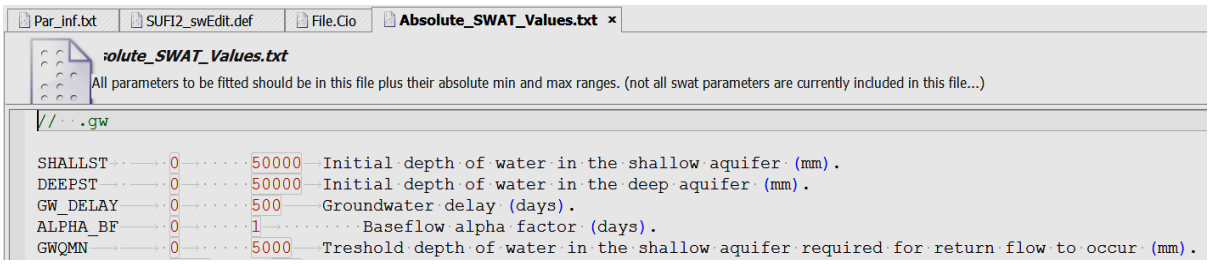
```
Output Information:
               0     | IPRINT: print code (month, day, year)
               3     | NYSKIP: number of years to skip output printing/summarization
               0     | ILOG: streamflow print code: 1=print log of streamflow
               0     | IPRP: print code for output.pst file: 1= print pesticide output
```

Miss-specifying the correct dates is the cause of biggest user error! Please note the following:

- In the above example, beginning year of SWAT simulation is 1987, end year is 2001
- There are 3 years of warm up period as indicated by NYSKIP. Therefore, SWAT output files contain data from 1990 to 2001. These dates are of interest to SWAT-CUP. So, in SWAT-CUP beginning year is 1990 and end year is 2001.
- Also note that SWAT-CUP requires the IDAF to be at the beginning of the year (always 1) and IDAL to go to the end of the year (365 or 366 for leap years). Therefore SWAT simulation should always be from the beginning to the end of the year. So your climate data must be from the beginning to the end of the year.

- **Absolute_SWAT_Values.txt**

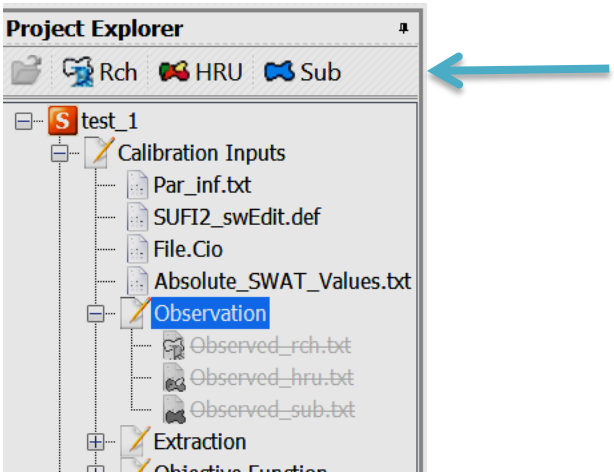
All parameters to be fitted should be in this file plus their absolute min and max ranges. Currently most, but perhaps not all parameters are included in this file. Simply add to it the parameters that don't exist. The SWAT_Edit.exe program, that replaces parameters in the SWAT files, does not allow parameters beyond this range into SWAT files.



etc.

6. Observation

Under **Observation** are three files that contain the observed variables. Observed variables correspond to the variables in output.rch, output.hru, and output.sub, output.res, and output.mgt files, although the latter two do not appear in SWAT-CUP.

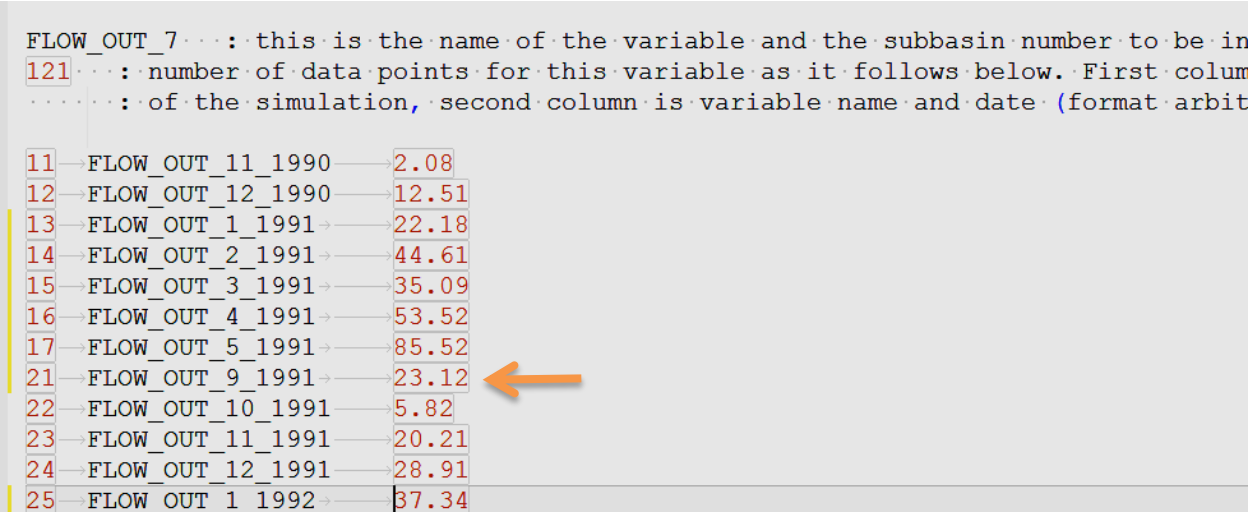


Initially, all options are deactivated. To activate you need to choose which SWAT file contains the simulated data (step 4 above). Variables from different files can be included to form a multi-component objective function. Simply **only edit** the file(s) that applies to your project and do not worry about the ones that don't. The format should be exactly as shown in the examples provided in the program. The three files Observed_rch.txt, Observed_hru.txt, and Observed_bsn.txt can be edited here, but observed_res.txt for reservoir data and observed_mgt.txt for crop yield are also available that could be edited directly in the .**SUFI2.IN** directory in the SWAT-CUP project directory.

Missing values

The format of the observation files are as shown in the examples provided. These files could easily be made in Excel and pasted here. In the observed files You may have missing data that can be accounted for as shown in the example files and explained below.

The first column has sequential numbers from the beginning of simulation time period. In the example below, the first 10 months are missing so the first column begins from 11. Also, months 18,19, and 20 are missing.



The second column has an "arbitrary format" **but it should be one connected string**. Here, it is showing the variable name, month, and year. Third column is the variable value.

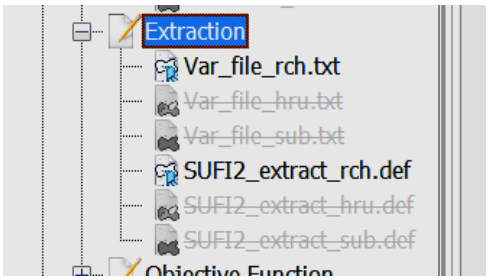
If base flow is separated, and dynamic base flow is used, then a forth column indicating the base flow should also be added. The example of an observation file with base flow is given in observed+.txt in **.\SUFI2.IN** directory.

All other observation_*.txt files have the same format. This file tells the extract programs of SWAT-CUP what to extract from the SWAT output files.

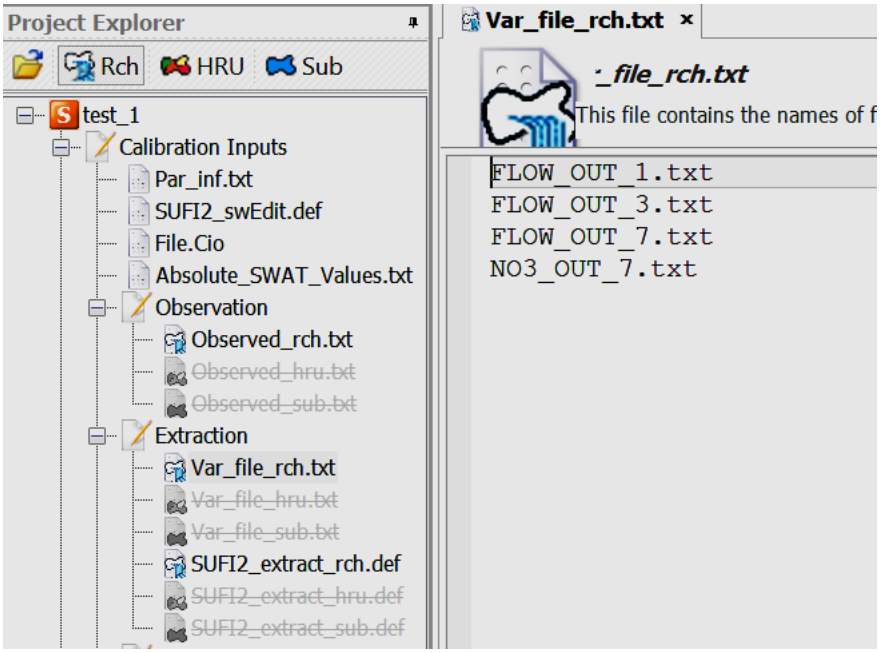
The observed_rch.txt files can contain many variables such as discharge, sediment, nitrate, etc. which appear in the SWAT output file output.hru. Simply use the same format for all variables as shown in the examples. Also, for the variables name, be consistent in all SWAT-CUP files.

7. Extraction

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 only.



.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.



In the example provided, we have 4 measured variables, 3 discharges from subbasins 1,3,7, and 1 nitrate from subbasin 7. The extract files of SWAT-CUP extract the corresponding simulated data from output.rch file and write them to the files indicated here for every simulation.

Below is an example of SUFI2_extrcat_rch.def

SUFI2_extract_rch.def ×

SUFI2_extract_rch.def

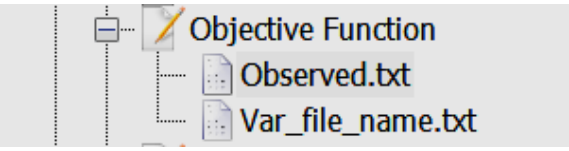
This file defines how variables should be extracted from the output.rch file.

```
output.rch          : swat output file name
2                   : number of variables to get
7  18               : variable column number(s) in the swat output file (as many as the above number)

20                  : total number of reaches (subbasins) in the project

3                   : number of reaches (subbasins) to get for the first variable
1  3  7             : reach (subbasin) numbers for the first variable (ordered)

1                   : number of reaches (subbasins) to get for the second variable (if not needed delete this and the next line))
7                   : reach (subbasin) numbers for the second variable (ordered)


1990                : beginning year of simulation not including the warm up period
2001                : end year of simulation

2                   : time step (1=daily, 2=monthly, 3=yearly)




// Remarks
```

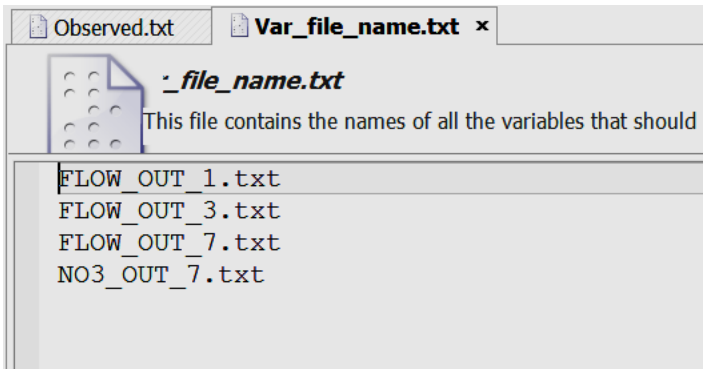
The file is self-explanatory. We are extracting 2 variables: discharge and nitrate. They are in columns 7 and 18 in the output.rch file. There are a total of 20 subbasins in the project. For discharge, we have flow measurements from subbasins 1, 3, and 7. For nitrate we have measurement from subbasin number 7 only. The beginning date "does not" include warm up period.

8. Objective function

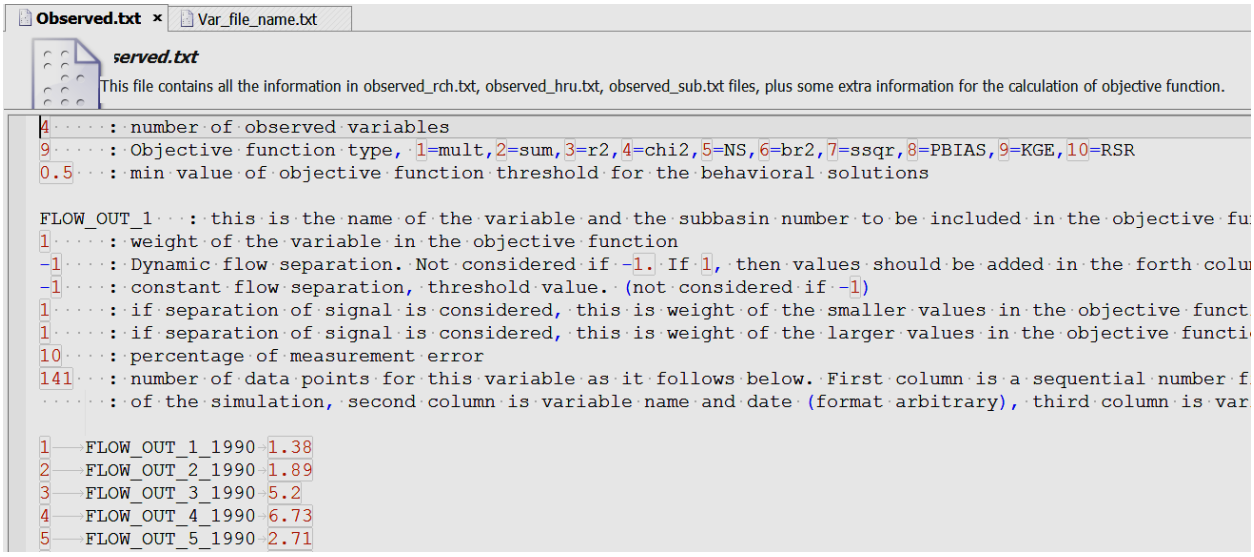
Next, **Objective Function** is defined. In this step two files **Observed.txt** and **Var_file_name.txt** should be edited. The **Observed.txt** file contains all the information in **observed_rch.txt**, **observed_hru.txt**, **observed_sub.txt** files, plus some extra information for the calculation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



-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.



-Observed.txt file also is quite self-explanatory.



In this example we have 4 variables. We have an option to choose from 10 different objective functions. Read the section on Objective Function below for more explanation of the functions.

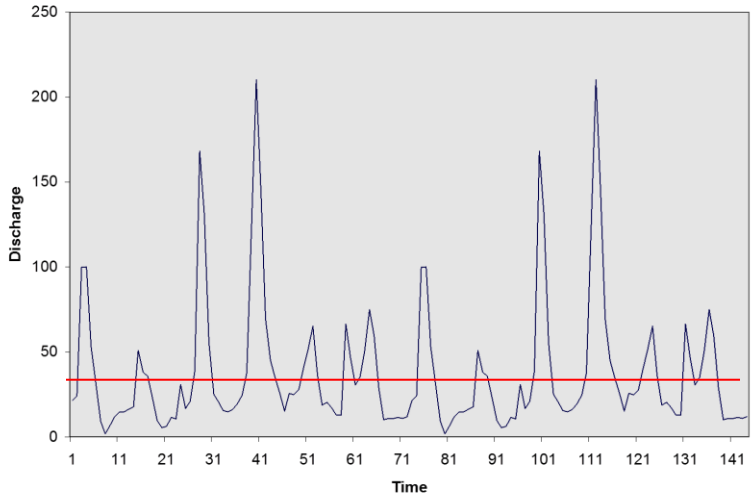
The third line is optional to run, but there should be a number here. If you express a threshold value here, then all simulations with objective function value better than the threshold are collected and the 95PPU calculated based on those simulations. Here, the SUFI2 becomes similar to GLUE. The *p-factor* and the *r-factor* for a given threshold may be different from the solution where threshold is not considered. Please note that the threshold value must correspond to the type of objective function being used.

Baseflow separation

Two options are provided to consider baseflow separation: *static* and *dynamic*. In the *static* case, a constant threshold value for baseflow is used. This values, divides the discharge signal into two parts. Values smaller than the threshold and values larger than the threshold are treated as two variables and carry two different weights. This is to ensure that, for example, base flow has the same values as the peak flows. Without this division, if you choose option 2 for objective function, i.e., mean square error (see objective function section below), then the small flows will not have much effect on the

optimization. Hence, peak flow will dominate the processes. With the static threshold option, small flows can be given a larger weight in the objective function so that they have almost the same contribution to the objective function as the peak flows. The base flow separation is most effective when option 2 is chosen for objective function. Separating the base flow does not become very critical if R^2 or bR^2 is used for objective function.

Threshold=35 →



To not use this option, simply set constant flow separation threshold to a negative value (say -1 for a variable that is always positive) and the weights for smaller and larger flows to 1.

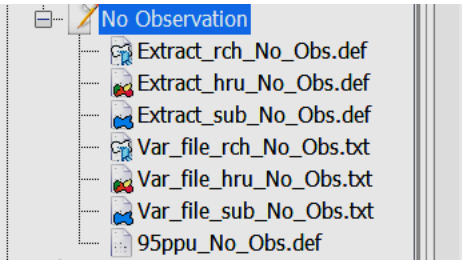
In the *dynamic* case, a flow separation program should be used to calculate the base flow. Both observed flow and baseflow must then appear in the **observed.txt** file as two separate columns, column 3 and column 4, respectively, as shown in the **observed+.txt** example file in **.\SUFI2.IN** directory.

Percentage of measured error

This value refers to the measurement error. A default value of 10% is specified, but the users can change this based on their knowledge. This value is reasonable for flow, but should be higher for other variables such as sediment and nitrate, etc.

9. No_Observations

The **No_Observation** section is designed for the extraction and visualization of uncertainties for the variables for which we have no observation, but would like to see how they are simulated such as various nutrient loads, or soil moisture, ET, etc. The **.txt** files are inactive.



The **.def** files have more or less the same format as the **Extraction** section.

Extract_rch_No_Obs.def **Extract_hru_No_Obs.def** **Extract_sub_No_Obs.def**

```
SUFI2          : SWAT-CUP program: SUFI2, GLUE, ParaSol, PSO, MCMC
output.rch     : swat output file name
3              : number of variables to get (such as: discharge, sediment, ET etc.)
7  8  11       : variable column number(s) in the swat output file (as many as the number above)
R-FLOW  R-EVAP  R-SED_OUT      :Names of variables

20             : total number of subbasins in the project

2              : number of subbasins to get for the first variable
3 5            : subbasin numbers for the first variable. Write "All" if equal to the total number of HRUs

20             : number of subbasins to get for the second variable (if not needed delete this and the next line)
ALL            : subbasin numbers for the second variable. Write "All" if equal to the total number of HRUs

3              : number of subbasins to get for the third variable (if not needed delete this and the next line)
1 2 3          : subbasin numbers for the third variable. Write "All" if equal to the total number of HRUs


1990           : beginning year of simulation not including warm up period
2001           : end year of simulation

2              : time step (1=daily,2=monthly, 3=yearly)
```

This files is also self-explanatory. The number of variables to get, column numbers (sequential), and representative variable names are specified (R- is used here to indicate these are from **output.rch** SWAT file) in rows 3-5. These names are used to build files where simulated values are collected for all simulations. Then total number of subbasins in the project is specified.

For each variable, we identify the number of subbasins to get, and the subbasin number(s). If we want to get all subbasin values, for example for plotting maps, then simply indicate ALL. This followed by beginning and end year of simulation. Again, beginning year of simulation should not include warm up period.

-95ppu_No_Obs.def

Finally, for NO_Observation option we need to edit the **95ppu_No_Obs.def** file. This is a file used for the calculation of the 95ppu for the extracted variables with no observation.

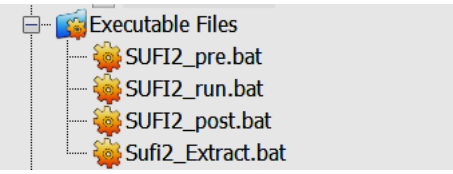
95ppu_No_Obs.def

```
SUFI2      :name of optimization program: SUFI2, Para_Sol, GLUE, PSO,
25         : number of variables to calculate 95PPU for without any ob

 :variable names, which should be the same as simulation file names:
R-FLOW_3.txt
R-FLOW_5.txt
R-EVAP_1.txt
R-EVAP_2.txt
.......
.......
R-EVAP_19.txt
R-EVAP_20.txt
R-SED_1.txt
R-SED_2.txt
R-SED_3.txt

144            : number of data points or simulation time steps
```

This file again is quite self-explanatory. The number of variables for which 95PPU is to be calculated is given in the second row. Variables' names are then provided. These names should be exactly the same as the ones given in the **.def** file(s). Finally, the number of simulation time steps are given. For 12 years of monthly simulation this would be 144.

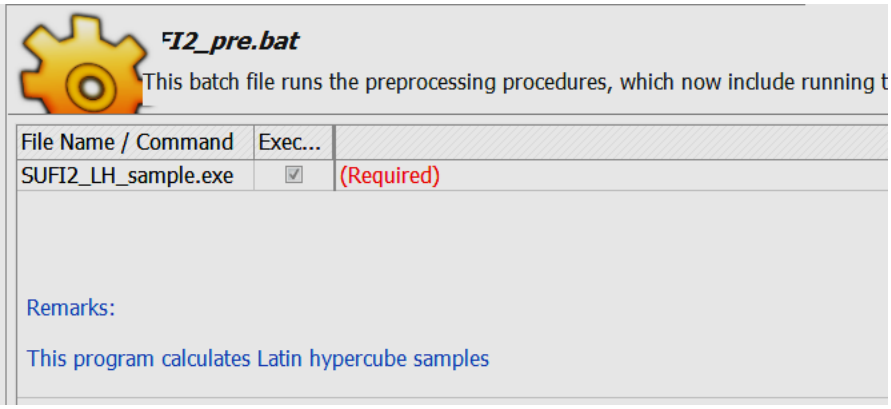
10. Executables

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

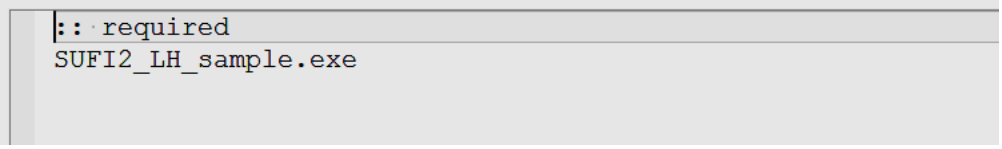


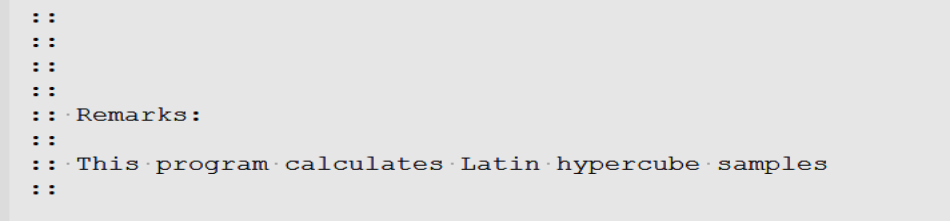
-SUFI2_pre.bat

This batch file runs the pre-processing procedures. It include running the Latin hypercube sampling program. This batch file usually does not need to be edited.



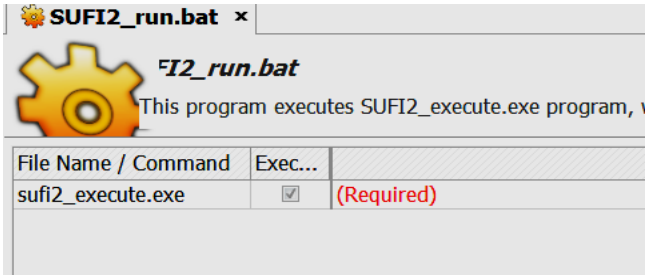
Note that many files at the end have Form View and Text View. In the Text View you have the text file, which appears in you SWAZ-CUP project directory. You can easily edit this text file as needed with the same format as shown.





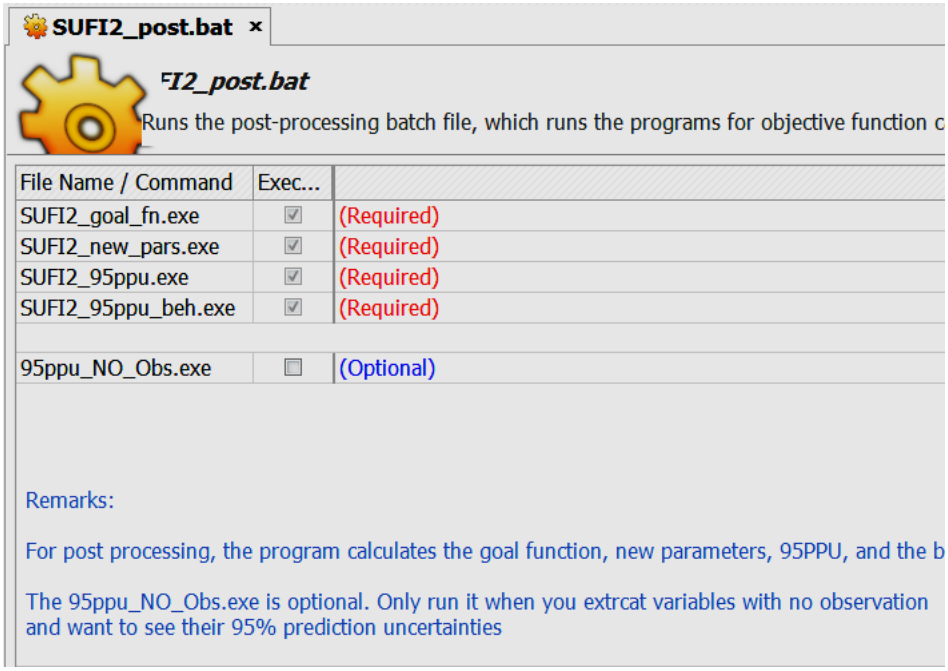
-SUFI2_run.bat

This program executes **SUFI2_execute.exe** program, which runs the **SWAT_Edit.exe**, extraction batch files, as well as **SWAT.exe**.



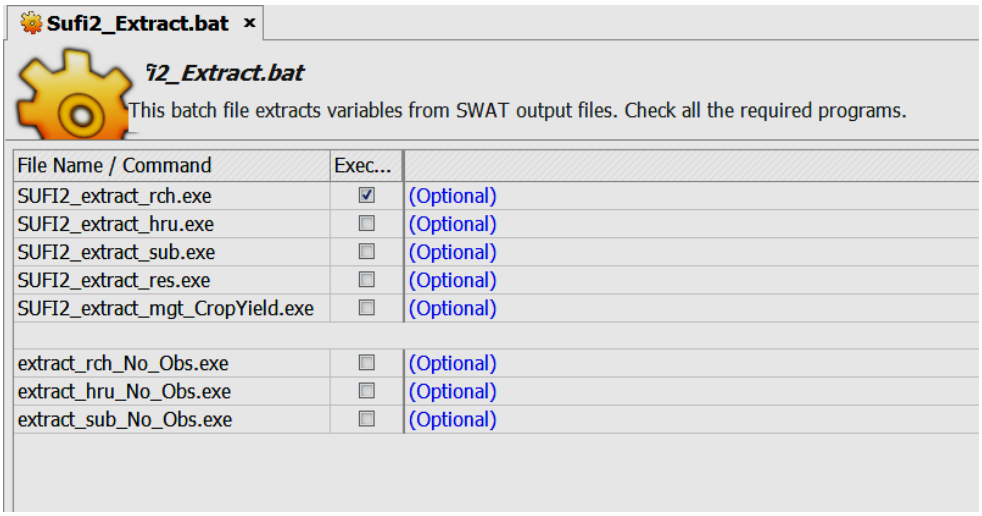
-SUFI2_post.bat

- 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 (optional). In the Text Form one can uncheck a program if it is not needed to run.



-SUFI2_Extract.bat

- This batch file contains the names of all the extract programs with or without observations. Currently 8 programs are supported. This file must be edited and the programs that are not desired to run should be "remarked" or "unchecked" as shown below:



11. Before Calibration

At this point the input files are complete and the project is ready to be calibrated. But before starting an iteration, you need to make sure that the model you have built in feasible initially. So, you should make a first run with the initial model structure and initial model parameters.

To check the initial model with SWAT-CUP do the following:

1-In Par_inf, put the number of simulations and the number of parameters to 1
2-In SUFI2_swEdit put the beginning and the ending simulation also to 1

3-Set up a dummy parameter such as:

r__CN2.mgt 0 0 (this does not change anything)

in Par_Inf.txt.

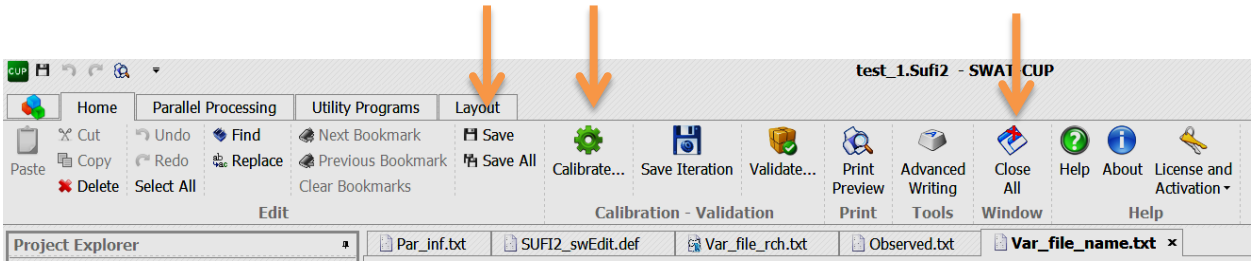
4-Then execute in order: Pre, Run, and Post processing.

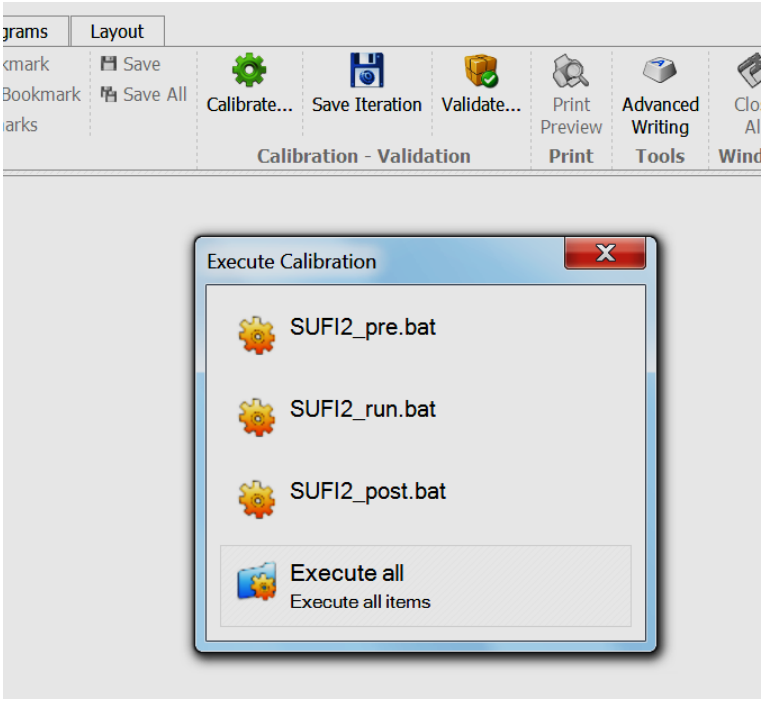
Now look at the 95PPU result of your default or initial model run. If simulations and observations are too different, then you need to take a closer look at your swat model, including rainfall, etc. Else, look at the calibration protocol in (http://www.sciencedirect.com/science/article/pii/S0022169415001985) to adjust the parameter in a way as to achieve the best simulation result at each observed outlet.

For this initial simulation, you should also look at the output.std file to make sure the overall watershed flow components are correct or not.

12. Calibration

Next, after editing all the input files, perform “Save All” and “Close All” tasks. The run the programs in the **Calibration** window in the order that they appear. In this section three steps are performed:



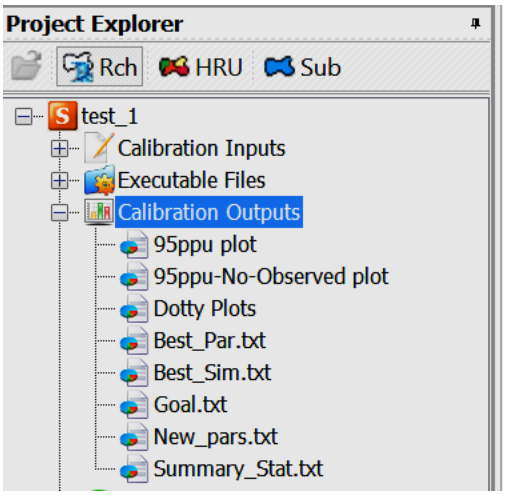


i) **Sufi2_pre.bat** - This command runs the **Sufi2_pre.bat** file. This file must be run before the start of every new iteration.

ii) **SUFI2_run.bat** - This command executes the run batch file.

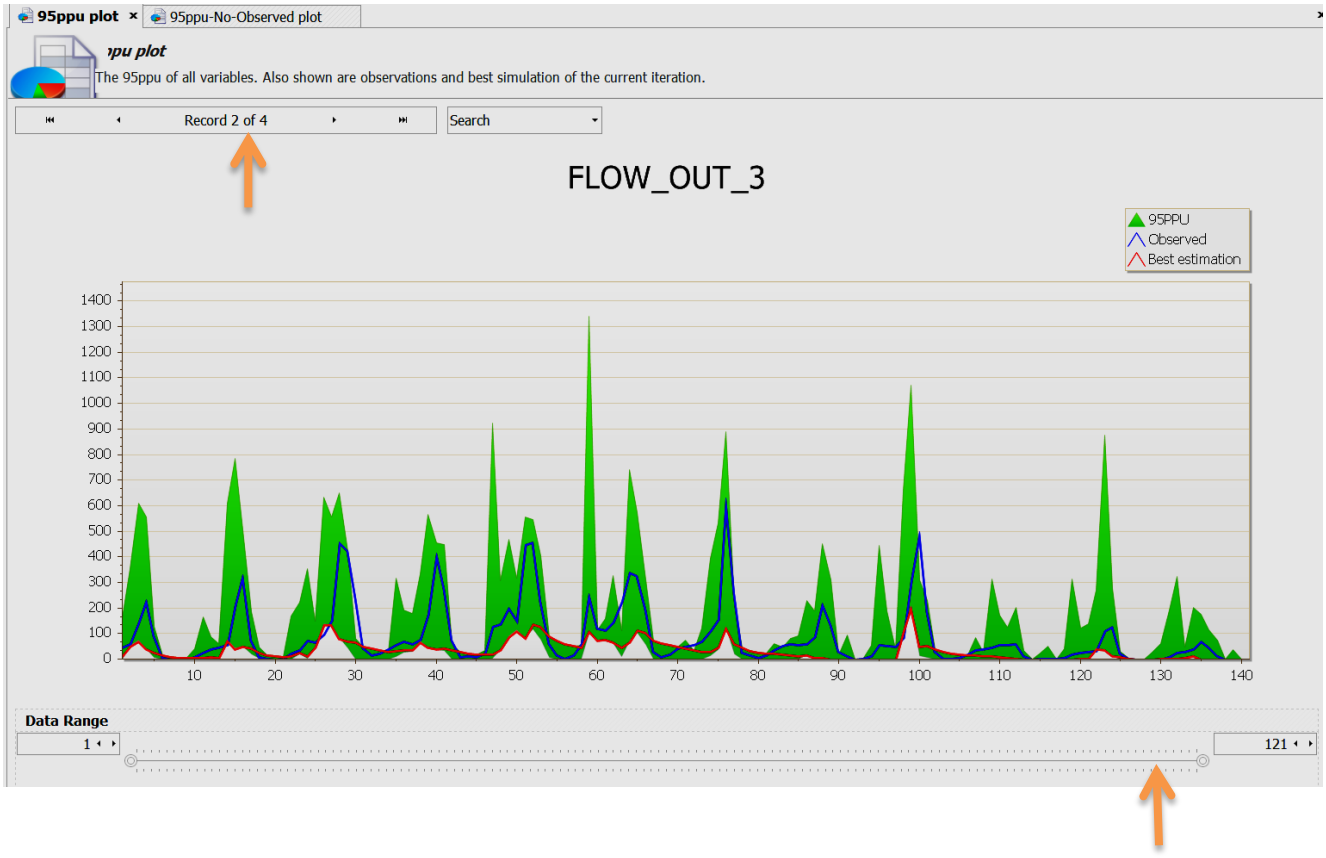
iii) **SUFI2_post.bat** - After all simulations are finished, this command executes the post processing batch file described above.

13. Calibration Outputs

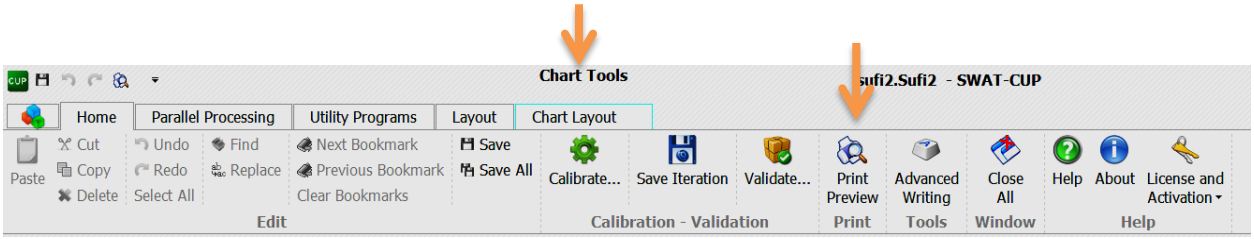


-95ppu plot

This command shows the 95ppu of all variables. Also shown are observations and best simulation of the current iteration. Please **Note** that the best simulation is only shown for historic reason. The solution to the calibration at this stage is the 95PPU graph and the parameter ranges that were used to generate it. Note the features with arrow where you can change the variables as well as zooming the hydrograph.

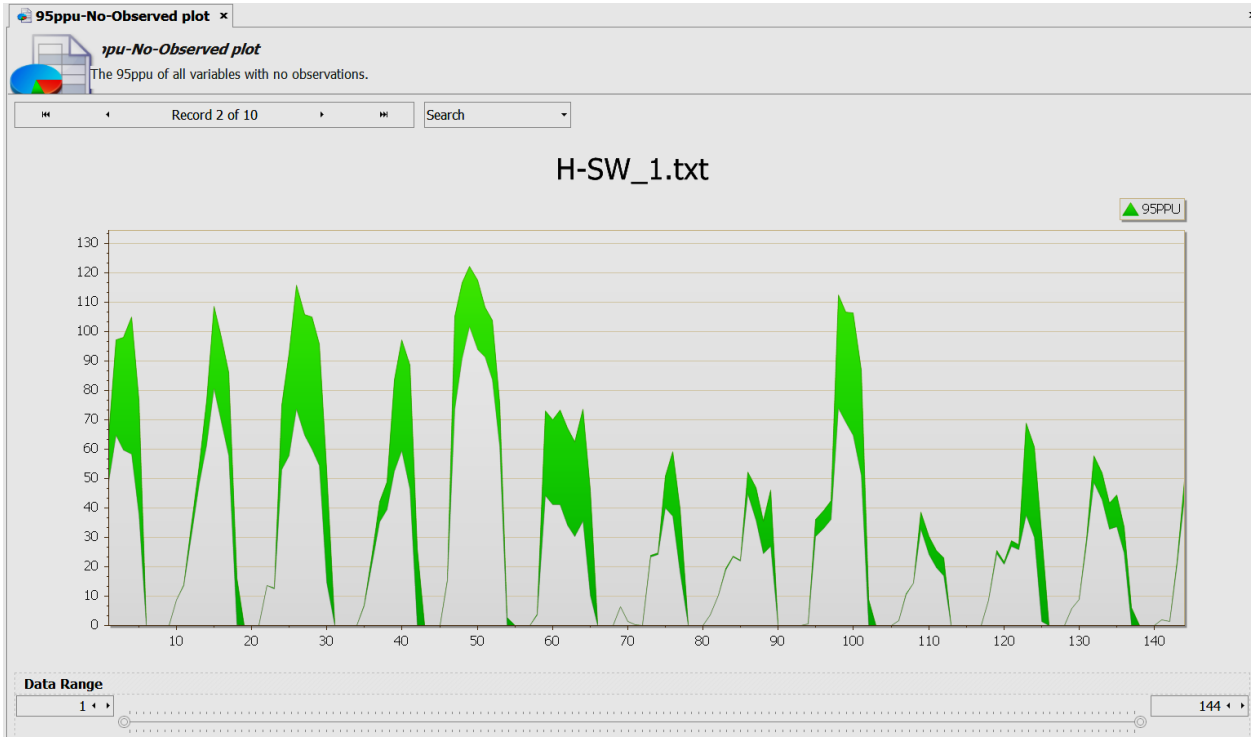


Also, please note the options given by “Chart Layout” and “Print Preview”



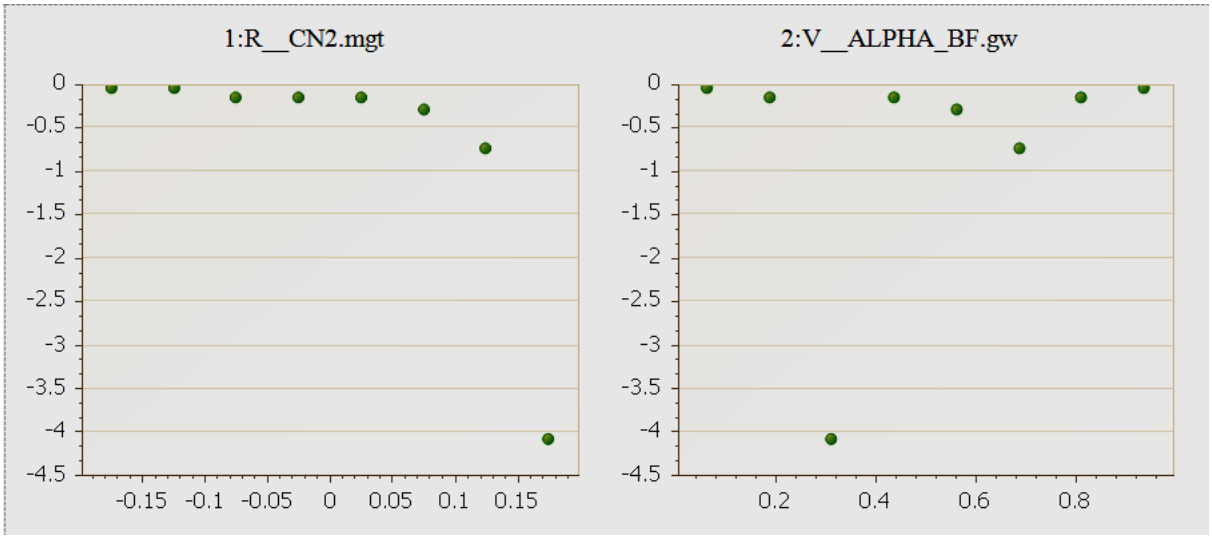
-95ppu-No_Observed plot

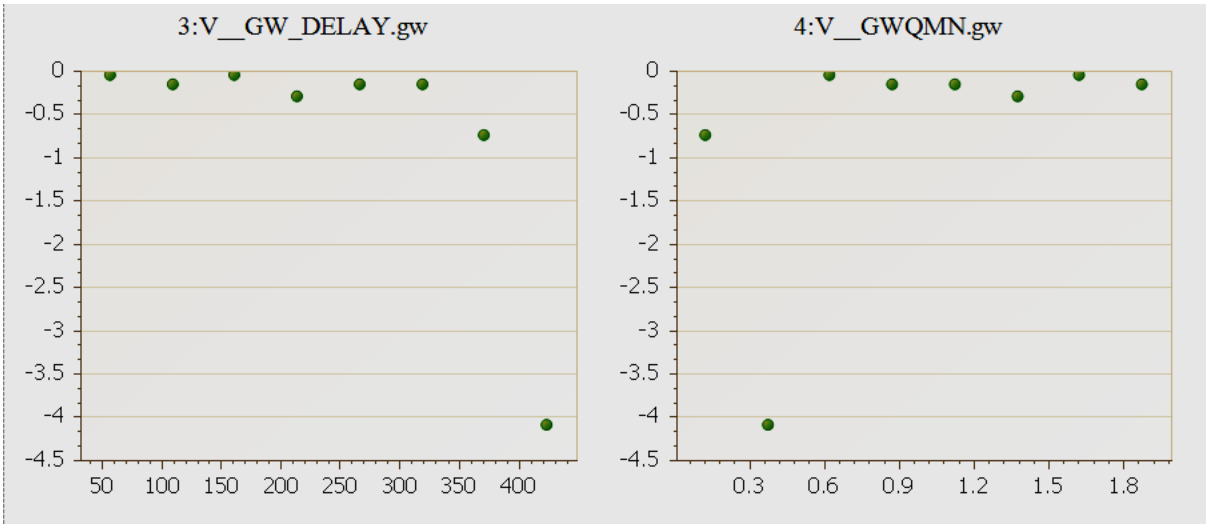
This command shows the 95ppu of all variables with no observations. Here you only see the uncertainty in simulation of Soil Moisture, a variable for which we have no observation.



-Dotty Plots

This command shows the dotty plots of all parameters. These are plots of parameter values or relative changes versus 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. In the following figure you see a nice trend for CN2 as it increases. Objective function is Nash-Sutcliffe (NS). Clearly CN2 is a sensitive parameter and its best fitting values are less than -0.1 in relative change (r__). But ALPHA_BF does not appear to be sensitive as the value of objective function doesn’t really change. GW_DELAY also is not very sensitive, but its value should probably not be above 300, GWQMN also not very sensitive, but it should probably be somewhere above 0.6. More about sensitivity later.





-Best_Par.txt

This file shows the "best parameter" values as well as their ranges. These are the parameters, which gave the best objective function value in the current iteration. Again, I like to emphasize that the best parameter really does not mean very much as the next objective function value may not be statistically not too different from the best one. The parameter ranges are the solution for this iteration.

```
Goal_type= Nash_Sutcliff    Best_sim_no= 7    Best_goal = -4.323317e-002

 Parameter_Name          Fitted_Value     Min_value      Max_value
1:R__CN2.mgt              -0.175000      -0.200000       0.200000
2:V__ALPHA_BF.gw           0.062500       0.000000       1.000000
3:V__GW_DELAY.gw         161.250000      30.000000     450.000000
4:V__GWQMN.gw              0.625000       0.000000       2.000000
```

-Best_Sim.txt

This file shows the best simulated values for all the variables used in the objective function. Both observed and simulated values are given so that they could easily be plotted with other softwares as desired.

```
FLOW_OUT_1
observed        simulated
1.3800          0.1293
1.8900          0.9779
5.2000          0.1027
```

……

```
FLOW_OUT_3
observed        simulated
46.2700         14.2100
59.3600         51.4300
143.5500        70.7800
```

……

```
NO3_OUT_7
observed        simulated
17000.0000      4596.0000
510000.0000     28490.0000
56000.0000      397.5000
```

-Goal.txt

This file shows the value of all parameter sets for the simulations performed as well as the value of the goal function in the last column. This file is used late to calculate the so-called “global sensitivity”.

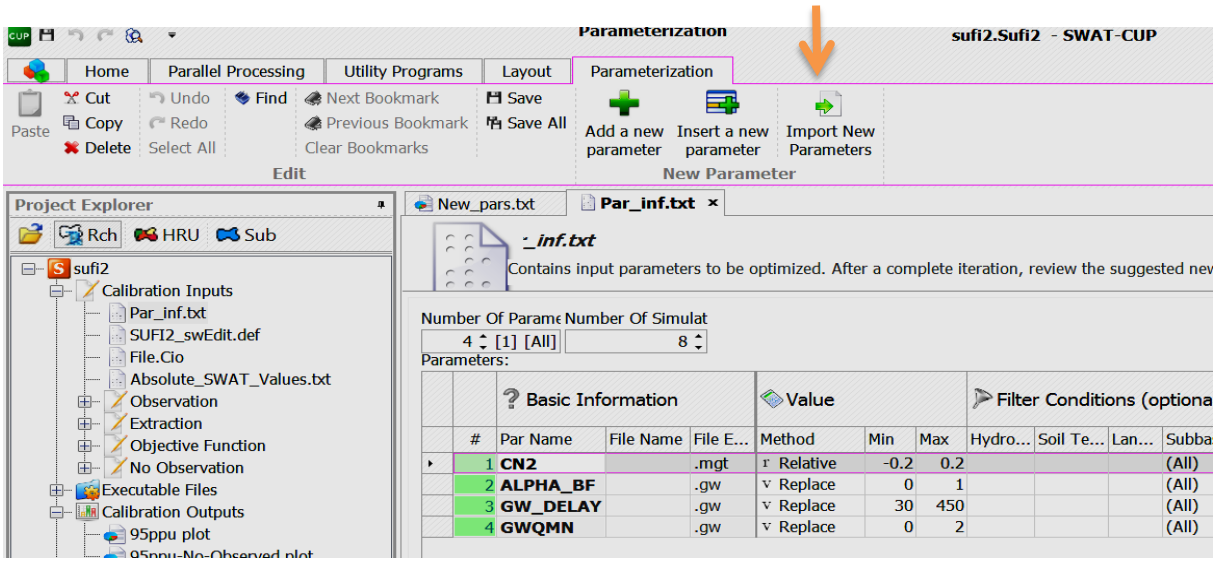
```
no_pars=  4
no_Sims=  8
type_of_goal_fn= Nash_Sutcliff
Sim_No. 1:R__CN2.mgt   2:V__ALPHA_BF.gw   3:V__GW_DELAY.gw   4:V__GWQMN.gw   goal_value
1        0.0250         0.8125            108.7500          0.8750          -0.148754
2        0.0750         0.5625            213.7500          1.3750          -0.291169
3        0.1750         0.3125            423.7500          0.3750          -4.092258
4        0.1250         0.6875            371.2500          0.1250          -0.741793
5       -0.0750         0.4375            318.7500          1.1250          -0.157484
6       -0.1250         0.9375            56.2500           1.6250          -0.048107
7       -0.1750         0.0625            161.2500          0.6250          -0.043233
8       -0.0250         0.1875            266.2500          1.8750          -0.153588
```

-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, or alternatively, the “Import New Parameters” could be used to copy new parameters into par_inf.txt file. The new parameters should be checked for possible unreasonable values (e.g., negative hydraulic conductivity,

etc.). These suggested parameter ranges should be manually corrected and if desired directed to a certain range by the user in case of available information or knowledge of the system.

par_no	par_name	new_min	new_max
	r__CN2.mgt	-0.486369	0.136369
	v__ALPHA_BF.gw	-0.494362	0.619362
	v__GW_DELAY.gw	-111.978348	434.478333
	v__GWQMN.gw	-0.249130	1.499130



-Summary_Stat

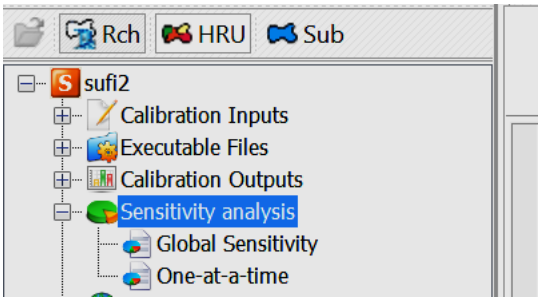
This file has the statistics of comparing observed data with the simulation band through *p-factor* and *r-factor* and the best simulation of the current iteration by using R^2, NS, bR^2, MSE, SSQR, PBIAS, KGE, RSR, and VOL_FR. The mean and standard deviation of the observed and simulated variables are also given at the end. For definition of these functions see the section on objective functions. Also shown is the goal function type, best simulation number of the current iteration, and the best value of the objective function for the current run on the top.

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= Nash_Sutcliff    No_sims= 8     Best_sim_no= 1    Best_goal = -5.415826e-002

Variable    p-factor  r-factor  R2    NS     bR2      MSE       SSQR       PBIAS  KGE   RSR   MNS   VOL_FR  ---  Mean_sim(Mean_obs)    StdDev_sim(StdDev_obs)
FLOW_OUT_1  0.50      0.59      0.10  -0.18  0.0030   3.7e+001  3.3e+001   87.5   -0.43 1.09  0.16  7.97          0.39(3.08)            0.57(5.62)
FLOW_OUT_3  0.79      1.54      0.33  0.10   0.0704   1.3e+004  8.4e+003   57.5   0.05  0.95  0.27  2.35          37.78(88.87)          43.51(119.08)
FLOW_OUT_7  0.62      2.03      0.03  -0.17  0.0024   7.4e+003  1.8e+003   31.6   -0.00 1.08  0.14  1.46          40.40(59.03)          43.07(79.74)
NO3_OUT_7   0.58      1.37      0.96  0.02   0.0793   1.9e+010  1.9e+010   92.4   -0.30 0.99  0.34  13.15         4171.07(54860.80)     11689.00(138810.14)


---- Results for behavioral parameters ---
Behavioral threshold= -2.500000
Number of behavioral simulations = 7

Variable    p-factor  r-factor  R2    NS     bR2      MSE       SSQR       PBIAS  KGE   RSR   MNS   VOL_FR  ---  Mean_sim(Mean_obs)    StdDev_sim(StdDev_obs)
FLOW_OUT_1  0.23      0.23      0.10  -0.18  0.0030   3.7e+001  3.3e+001   87.5   -0.43 1.09  0.00  7.97          0.39(3.08)            0.57(5.62)
FLOW_OUT_3  0.49      0.74      0.33  0.10   0.0704   1.3e+004  8.4e+003   57.5   0.05  0.95  0.00  2.35          37.78(88.87)          43.51(119.08)
FLOW_OUT_7  0.55      1.02      0.03  -0.17  0.0024   7.4e+003  1.8e+003   31.6   -0.00 1.08  0.00  1.46          40.40(59.03)          43.07(79.74)
NO3_OUT_7   0.50      0.28      0.96  0.02   0.0793   1.9e+010  1.9e+010   92.4   -0.30 0.99  0.00  13.15         4171.07(54860.80)     11689.00(138810.14)
```

14. Sensitivity analysis

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

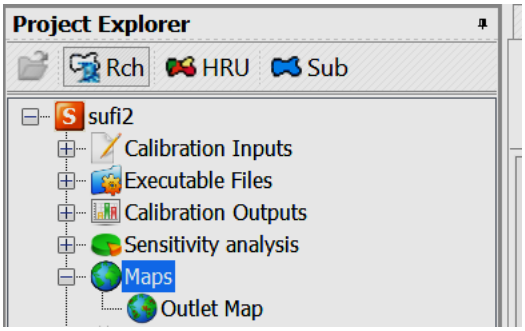


Global sensitivity analysis can be performed after an iteration. One-at-a-time sensitivity is performed for one parameter at a time only. The procedure is explained in the next section.

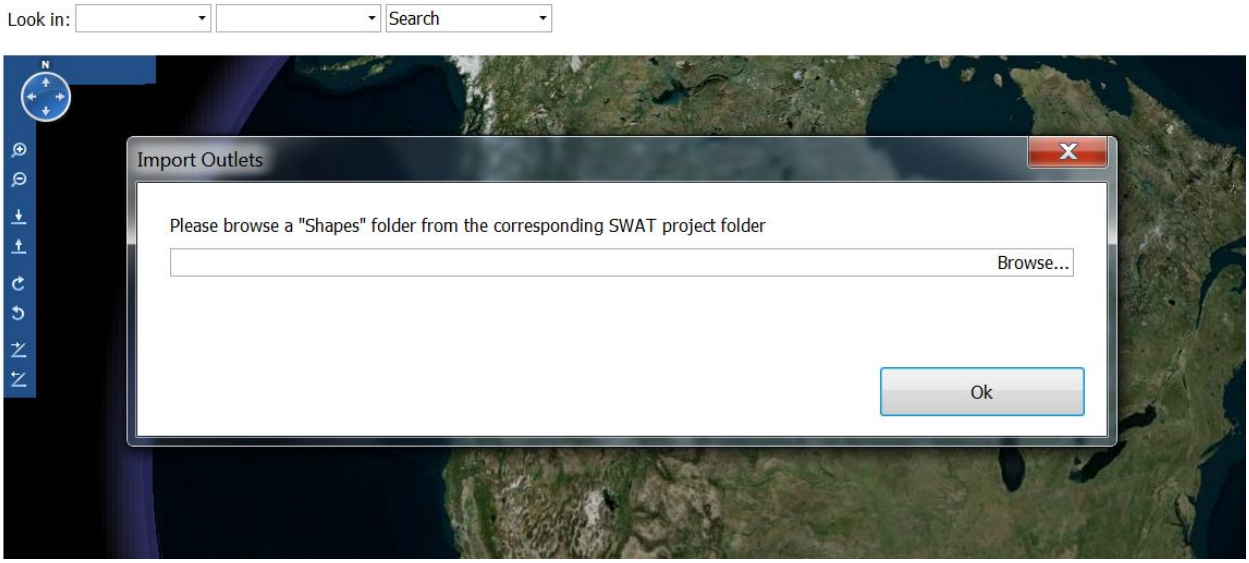
15. Maps

The Maps module enables visualization of the outlets, rivers, and subbasins. The Bing map is used to project the location of outlets, rivers, and subbasins on the actual map of the world. This option is useful in identifying if the rivers are correctly digitized or not, if outlets are in their correct location or not, if there are dams/reservoirs, areas of high mountains with snow and glaciers, wetlands, intensive agricultural areas or cities with high levels of water management, etc. These information are critical for proper calibration of a model.

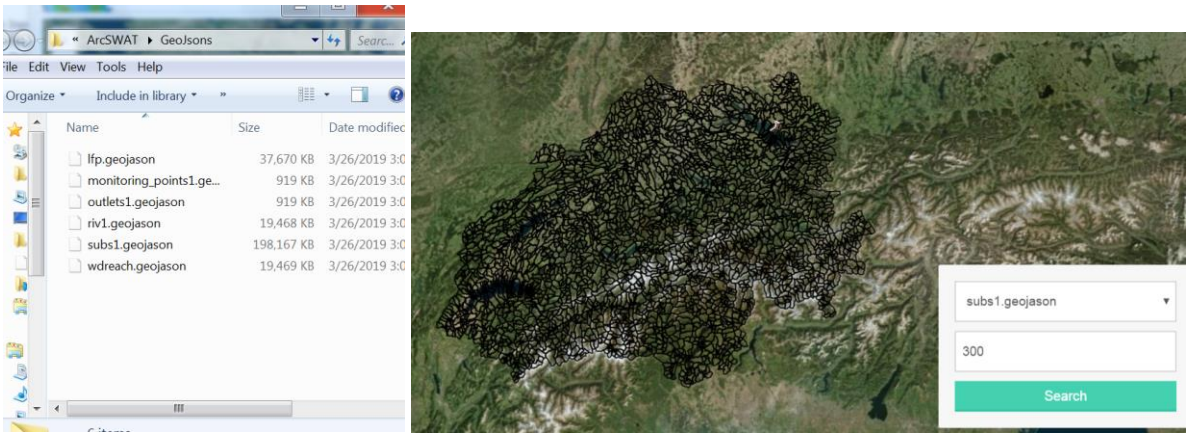
First, by invoking the Outlet Map as shown below the Bing map will be activated.



In the second step, invoke the “Import Outlets” from the main menu. The program asks for the ArcSWAT project’s **Shapes** folder.



Upon locating the Shape file folder in the ArcSWAT project, “...\Watershed\Shapes”, a folder “geojason” is created where shape files are reformatted as geojason files. These files can then be dragged to the Bing from the .Sufi2.SwatCup\ArcSWAT\GeoJsons folder to be visualized.

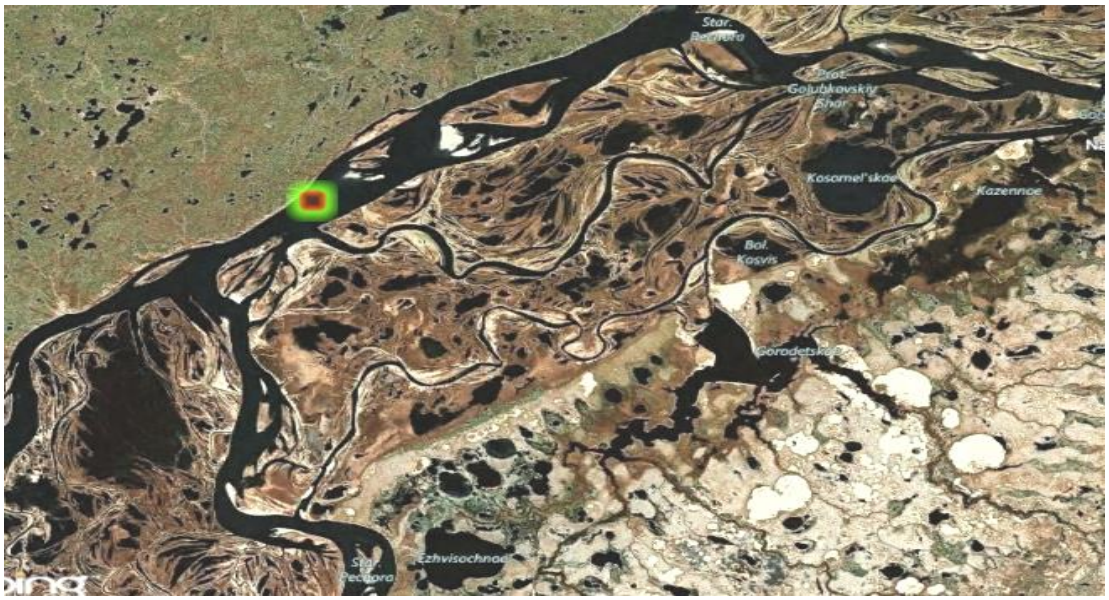


Some examples of the situation that could be detected by Maps option.

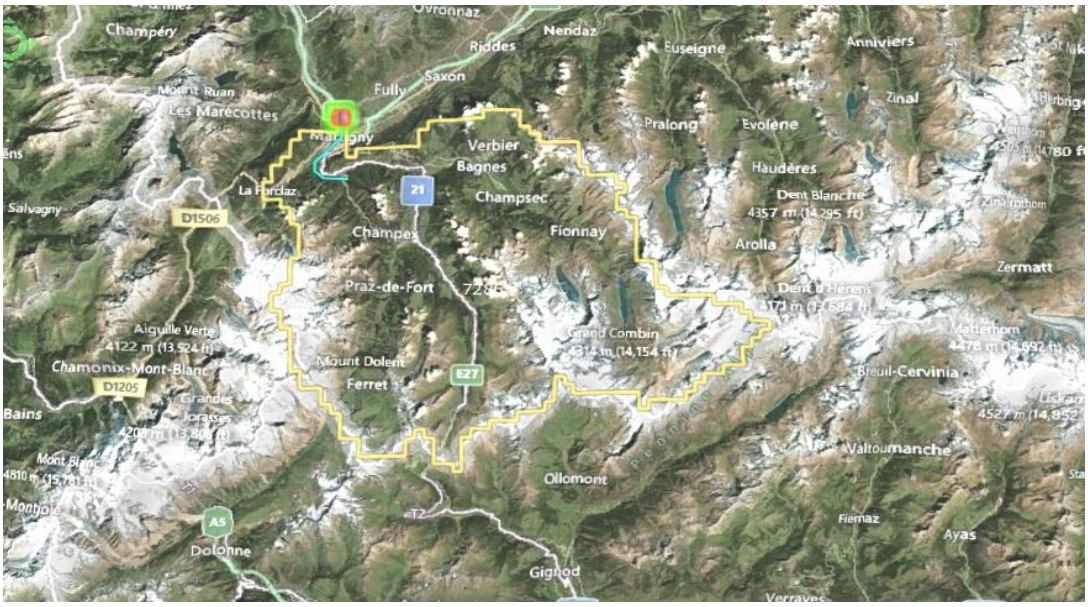
a) Wrong positioning of an outlet. Below an outlet is placed on the Viar river, which is a tributary of the main Rio Guadalquivir river in Spain. The red-green symbol indicates the SWAT positioning of the outlet due to given imprecise coordinates. The actually location of the outlet is on the Rio River shown with the yellow circle. This is a common mistake and if not detected, will create a big headache for calibration, ultimately leading to wrong parameters.

b) The position of an outlet downstream of a dam on Inn River near Munich, Germany. SWAT cannot calibrate the flow in this outlet unless the reservoir is modeled or its discharge is known.

c) a complex river geometry on Pechora River near Golubovo in Russia. SWAT cannot be expected to simulate the flow in this outlet with high accuracy.

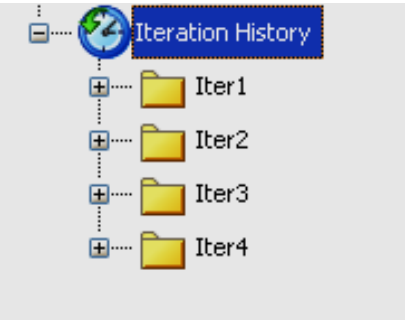


d) The flow in the outlet below is governed by glacier melt near Martigny in Switzerland. As glacier melt is currently not simulated in SWAT, some discrepancies between simulation and observed results should be expected when calibrating this outlet.



16. Iterations History

All iterations can be saved in the iteration history. This allows studying the progress to convergence.



After a complete iteration, review the suggested new parameters in the **new_pars.txt**, copy them to **par_inf.txt** and edit them as explained before, and make a new iteration. There are no hard rules as to when a calibration process can be terminated. But the process can stop when satisfactory statistics are achieved and there are no further improvements in the objective function value.

Parameterization in SWAT-CUP

The following scheme can be used to parameterize, or regionalize parameters of a watershed. 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 spatial parameter such as hydraulic conductivity, bulk density, or CN2 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 to lump them based on soil type, landuse type, location, slope, or a combination of these. They can then be calibrated using a single global modification term that can scale the initial estimates by a multiplicative, or an additive term. This leads to the following proposed parameter identifiers explained below.

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

Where

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

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

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

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

Note: that there are always two underscores __ after the identifier

<parname> = SWAT parameter name as it appears in the SWAT I/O manual or in the *Absolute_SWAT_Values.txt* file.

<ext> = SWAT file extension code for the file containing the parameter (e.g., .sol, .hru, .rte, etc.)

<hydrogrp> = (optional) soil hydrological group ('A','B','C' or 'D')

<soltext> = (optional) soil texture as it appears in the header line of SWAT input files

<landuse> = (optional) name of the landuse category as it appears in the header line of SWAT input files

<subbsn> = (optional) subbasin number(s) as it appears in the header line of SWAT input files

<slope>= (optional) slope as it appears in the header line of SWAT input files

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 after every previous specifications must be used, i.e., to specify only the subbasin we must write eight underscores after .crp 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, elevation, and slope. The parameters can be assigned and calibrated regionally, 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. Some examples of the parameterization scheme is as follows:

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__FRT_KG{9,1}.mgt	In these three examples, rotation 9, operation 1, and the rest are filters where , means AND
r__FRT_KG{9,1,PLANT_ID=12}.mgt	
r__FRT_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 and Temperature 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
v__MAXTEMP(1){1977001}.tmp1.tmp	(1) means column 1 in the tmp1.tmp file

	{1977001} specifies year and day
v__MAXTEMP(2){1977002-1977007}.tmp1.tmp	(2) means column 2 in the tmp1.tmp file from day 2 to day 7 in 1977
v__MINTEMP (){1977002-1977007}.tmp1.tmp	() means all columns in tmp1.tmp file

Specification of slope Parameters

Parameter identifiers	Description
v__SOL_K(1).sol_____0-10	K of layer 1 for HRUs with slope 0-10

Please note that brackets ( ) are used to distinguish layers in parameters that have many layers.

Also, please note that precipitation and temperature are also allowed to be used as fitting parameters. This option must be used with caution because fitting rainfall can make calibration of parameters irrelevant as rainfall is the single most important driving variable controlling the behavior of flow.

Objective Function Definition

In the observed.txt file, 10 different objective functions are currently allowed. **These include:**

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

This is a multiplicative form of the square error where Q, S, and N stand for variables (e.g., discharge, sediment, and nitrate), n is the number of observations, and m and s stand for measured and simulated. Sometimes the denominator is divided by 1000 to keep g small.

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

This is the summation form of the square error where Q, S, and N stand for variables (e.g., discharge, sediment, and nitrate), m and s stand for measured and simulated, n is the number of data points, and weights w's could be calculated as:

i) $w_j = 1 \Big/ n_j \sigma_j^{\,2}$

where $\sigma_j^{\,2}$ is the variance of the j^{th} measured variable (see Abbaspour, et al., 2001), or

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

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

Coefficient of determination R^2 where Q is a variable (e.g., discharge), and m and s stand for measured and simulated, i is the i^{th} measured or simulated data. If there are more than one variable, then the objective function is defined as:

$$g = \sum_j w_j R_j^2$$

Where w_j is the weight of j^{th} variable.

4=Chi2 Minimize: $$\chi^2 = \frac{\sum_i (Q_m - Q_s)_i^2}{\sigma_m^2}$$

where Q is a variable (e.g., discharge), and m and s stand for measured and simulated, respectively, and σ_m^2 is the variance of measured data. If there are more than one variable, then the objective function is calculate as:

$$g = \sum_j w_j \chi_j^2$$

Where w_j is the weight of j^{th} variable.

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

Nash-Sutcliffe (1970), where Q is a variable (e.g., discharge), and m and s stand for measured and simulated, respectively, and the bar stands for average. If there is more than one variable, then the objective function is defined as:

$$g = \sum_j w_j NS_j$$

Where w_j is the weight of j^{th} variable.

6=bR² Maximize: $$\phi = \begin{cases} |b| R^2 & if \quad |b| \le 1 \\ |b|^{-1} R^2 & if \quad |b| > 1 \end{cases}$$

Where Coefficient of determination R^2 is multiplied by the coefficient of the regression line between measured and simulated data, b. 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). If more than one variable, the objective function is expressed as (Krause et al., 2005):

in case of multiple variables, g is defined as:

$$g = \sum_j w_j \phi_j$$

Where w_j is the weight of j^{th} variable.

7=SSQR Minimize: $SSQR = \frac{1}{n}\sum_{i=1}^{n}\left[Q_{i,m} - Q_{i,s}\right]^2$

where Q is a variable (e.g., discharge), and m and s stand for measured and simulated, respectively. Here i represents the rank. The SSQR method aims at fitting the frequency distributions of the observed and the simulated series. After independent ranking of the measured and the simulated values (van Griensven and Bauwens, 2003):

in case of multiple variables, g is defined as:

$$g = \sum_j w_j SSQR_j$$

Where w_j is the weight of j^{th} variable.

8. PBIAS Minimize: $$PBIAS = 100 * \frac{\sum_{i=1}^{n}\left(Q_m - Q_s\right)_i}{\sum_{i=1}^{n} Q_{m,i}}$$

where Q is a variable (e.g., discharge), and m and s stand for measured and simulated, respectively. Percent bias measures the average tendency of the simulated data to be larger or smaller than the observations. The optimum value is zero, where low magnitude values indicate better simulations. Positive values indicate model underestimation and negative values indicate model over estimation (Gupta et al., 1999).

in case of multiple variables, g is defined as:

$$g = \sum_j w_j PBIAS_j$$

Where w_j is the weight of j^{th} variable.

9. KGE

Maximize: $KGE = 1 - \sqrt{(r-1)^2 + (\alpha-1)^2 + (\beta-1)^2}$

Where $\alpha = \frac{\sigma_s}{\sigma_m}$, and $\beta = \frac{\mu_s}{\mu_m}$, and r is the linear regression coefficient between simulated and measured variable, μ_s and μ_m are means od simulated and measured data, and σ_s and σ_m are the standard deviation of simulated and measured data. Kling–Gupta efficiency (Gupta et al., 2009).

in case of multiple variables, g is defined as:

$$g = \sum_j w_j KGE_j$$

Where w_j is the weight of j^{th} variable.

10. RSR

Minimize: $$RSR = \frac{\sqrt{\sum_{i=1}^{n} (Q_m - Q_s)_i^2}}{\sqrt{\sum_{i=1}^{n} (Q_{m,i} - \overline{Q}_m)^2}}$$

where Q is a variable (e.g., discharge), and m and s stand for measured and simulated, respectively. RSR standardizes the RMSE using the observation standard deviation. RSR is quite similar to Chi in 4. It varies from 0 to large positive values. The lower the RSR the better the model fit (Moriasi et al., 2007).

in case of multiple variables, g is defined as:

$$g = \sum_j w_j RSR_j$$

Where w_j is the weight of j^{th} variable.

11. MNS

Maximize: $$MNS = 1 - \frac{\sum_i |Q_m - Q_s|_i^p}{\sum_i |Q_{m,i} - \overline{Q}_m|_i^p}$$

Modified Nasch-Sutcliffe efficiency factor. If p=2, then this is simply NS as in 5 above. If p=1, the overestimation of a peak is reduced significantly. The modified form is reported to be more sensitive to significant over- or under-prediction than the square form. Increasing the value of p beyond 2 results in an increase in the sensitivity to high flows and could be used when only the high flows are of interest, e.g. for flood prediction (Krause et al., 2005)

NOTE: After an iteration, try changing 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 calibration solution.

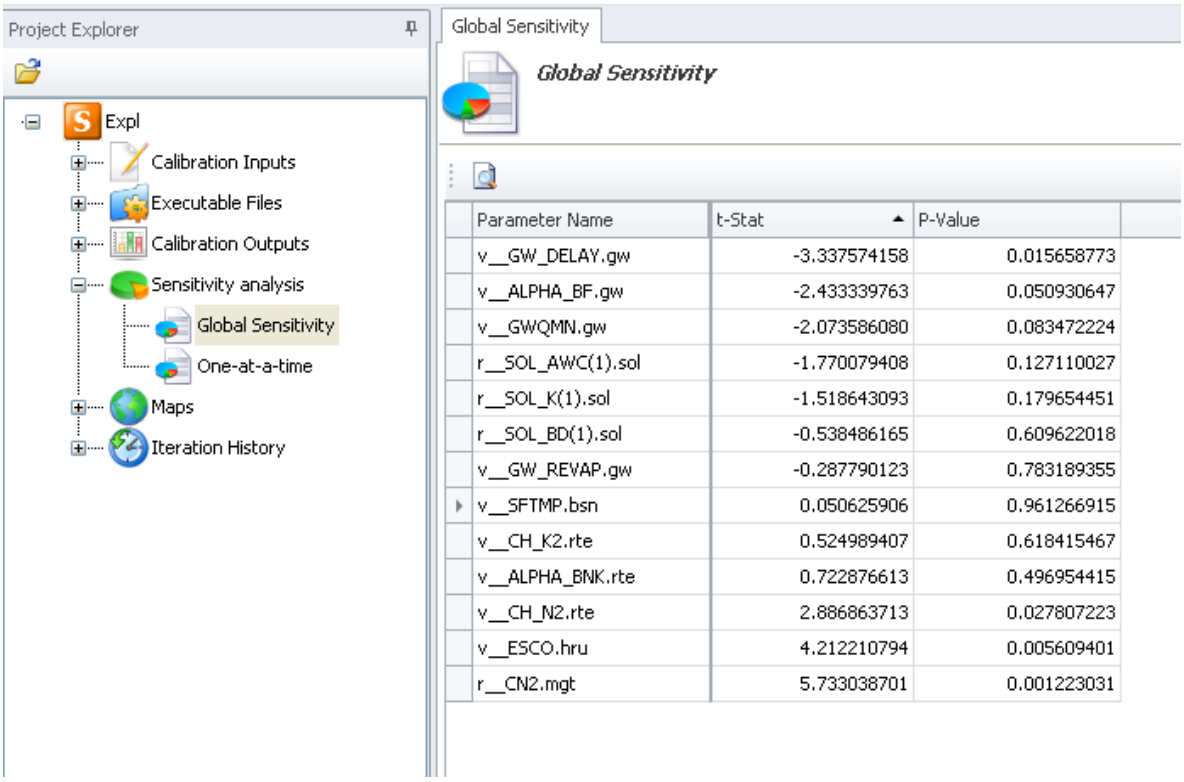
Sensitivity Analysis

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.txt):

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

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. In this analysis, the larger, in absolute value, the value of t-stat, and the smaller the p-value, the more sensitive the parameter. In the example below, CN2, ESCO, followed by GE_DELAY, CH_N2, and ALPHA_BF are the five most sensitive 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

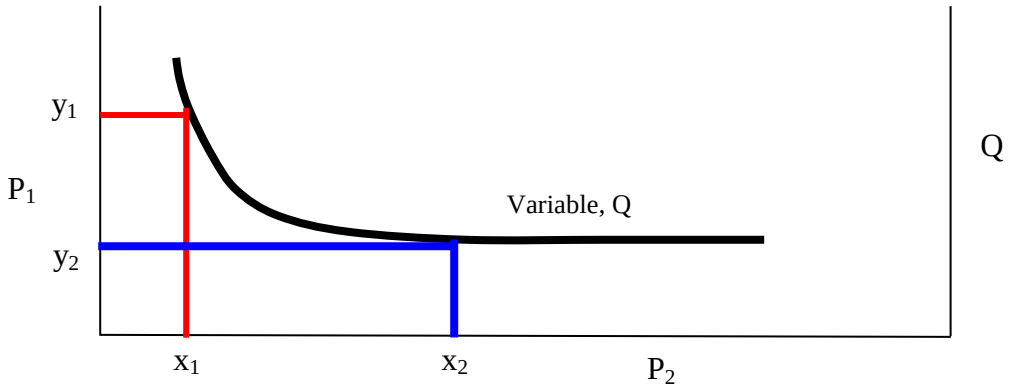
t-stat and p-value

A multiple regression analysis is used to get the statistics of parameter sensitivity. The *t-stat* is the *coefficient* of a parameter divided by its *standard error*. It is a measure of the precision with which the regression coefficient is measured. If a coefficient is "large" compared to its standard error, then it is probably different from 0 and the parameter is sensitive. What is "large"?

You could compare the *t-stat* of a parameter with the values in the *Student's t-distribution* table to determine the *p-value*, which is the number that you really need to be looking at. The *Student's t-distribution* (you find at the end of most statistics book) describes how the mean of a sample with a certain number of observations is expected to behave. The *p-value* for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low *p-value* (< 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low *p-value* is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable. Conversely, a larger *p-value* suggests that changes in the predictor are not associated with changes in the response. So that parameter is not very sensitive. A *p-value* of < 0.05 is the generally accepted point at which to reject the null hypothesis (i.e., the coefficient of that parameter is different from 0). With a *p-value* of 0.05, there is only a 5% chance that results you are seeing would have come up in a random distribution, so you can say with a 95% probability of being correct that the variable is having some effect.

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.

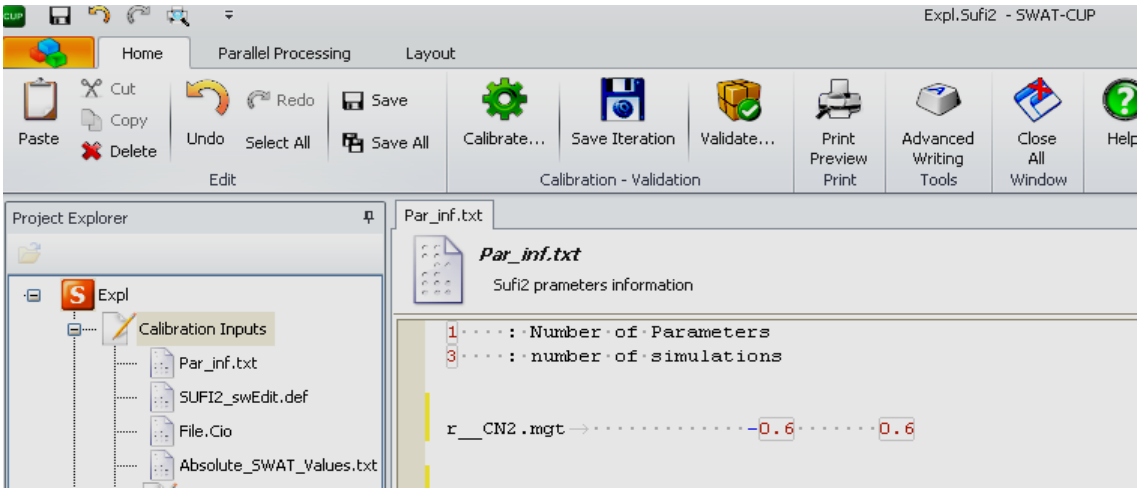


The above example illustrates this point. If value of parameter P_1 is kept constant at y_1, then small changes is parameter P_2 make significant changes in variable Q, indicating that P_2 is quite a sensitive parameter. While if the values of parameter P_1 is kept constant at y_2 value, then changes in parameter P_2 around x_2 will give the impression that P_2 is not a sensitive parameter as the variable does not change

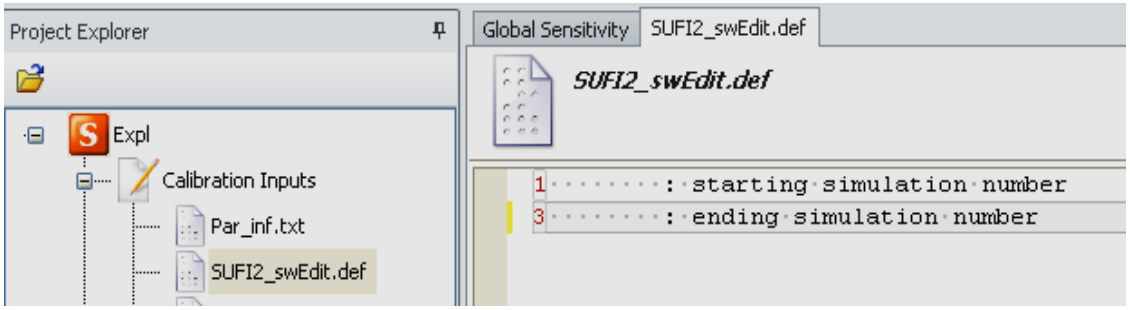
by much. Therefore, the values of the fixed parameters make a difference to the sensitivity of a changing parameter.

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.

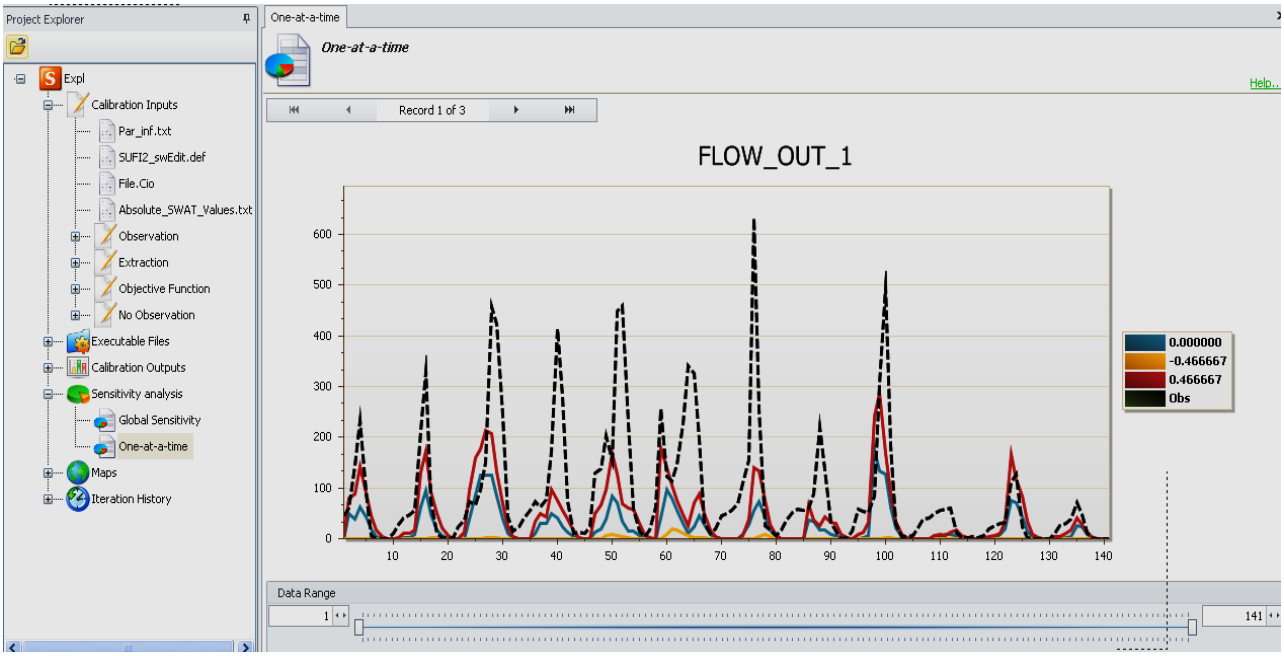


2- Then set the values of file SUFI2_swEdit.def as follows:



3- Finally perform the iteration 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 is sensitive and needs to have larger values.

NOTE: The users must be aware that the parameters in the SWAT files in the main SWAT-CUP project directory are always changing. Once you perform a sensitivity 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, or the initial model parameters that reside in the Backup directory.

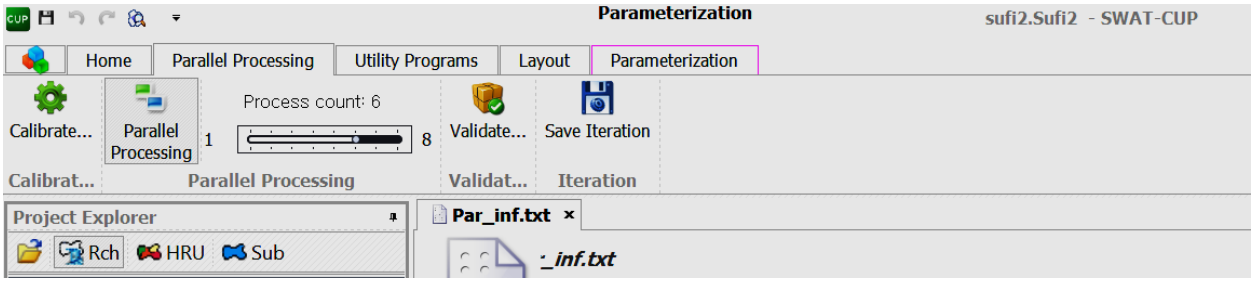
Parallel Processing

Parallel processing is a licensed product. Its function is to speed up the calibration process by parallelizing the runs in SUFI2. 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 so that 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 1000-run submission.

Now a days it is possible to build quite inexpensively a computer with 48 to 64 CPUs and more than 96 GB of RAM. Most SWAT models of any detail could be run on such machines without the need for cloud or grid computing (see Rouholahnejad, et al., 2012 for more detail).

Currently, 20 simulations are allowed to be made without the need for a license. To obtain a license follow the direction under license and activation and send the hardware ID, for the time being, to (neprach_sale@yahoo.com). After obtaining a license file by email, follow the activation process.

To run parallel processing, simply click the **Parallel Processing** button on the command bar. A new set of command icons appear. Press “parallel Processing” to see how many jobs can be submitted to your computer. Under “Process count” you can choose how many parallel jobs you want to submit 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 works as before.

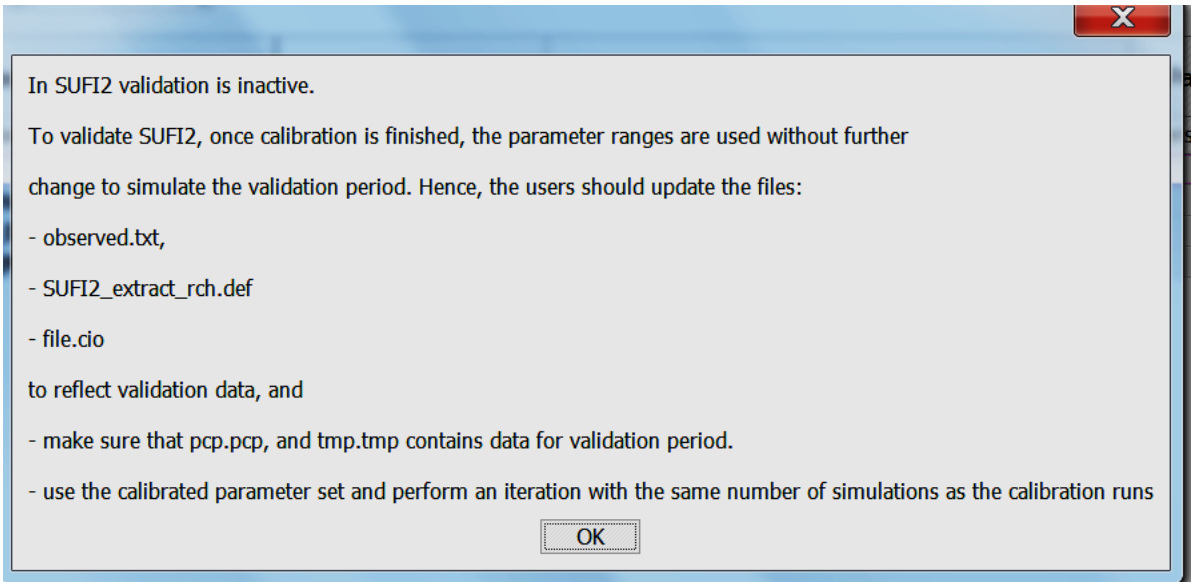


Validation in SUFI2

For validation, you should use the calibrated parameter ranges “without any further changes” and run an iteration (with the same number of simulations as you used for calibration).

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 (using the calibration button). The 95PPU and the Summary_stat file should reflect the validation results.

The validation key brings up the following menu, which explains the validation steps.



Example: We have discharge data from 1988 to 2012

-Make climate data from 1985 to 2012

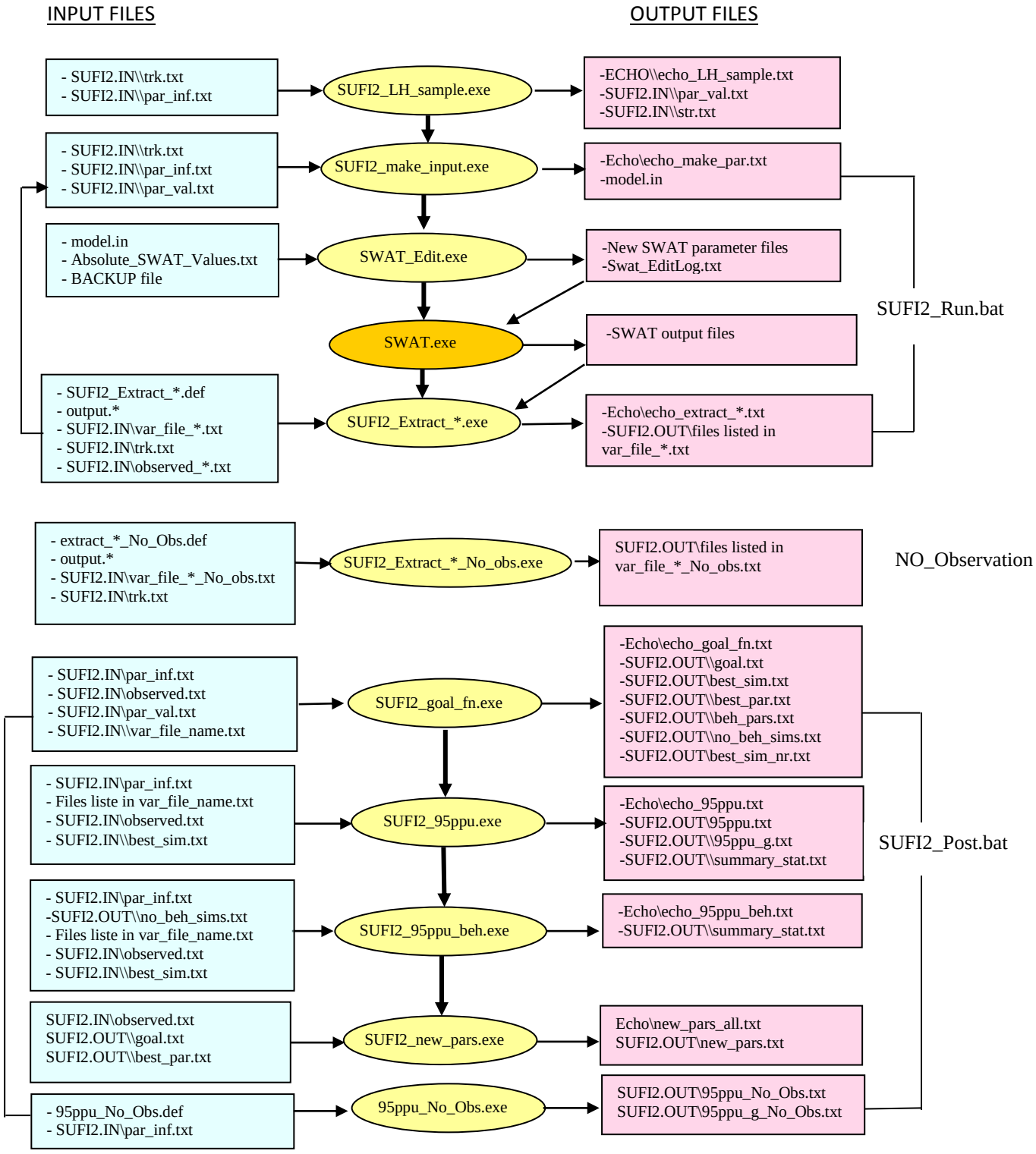
-Calibrate from 1985 to 2000 with the first three years as warm up period.

-Validate from 1998 to 2012 with the first three years as warm up period

For validation use the calibrated parameter ranges and do one iteration with as many simulations as the last calibration iteration.

The sequence of program execution

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



How to see the results of my initial model?

Before starting iterations in SWAT-CUP, you should check simulation of your initial model set up. It is assumed that some thought and investigation has gone into data collection and the best information is used to build the SWAT model. To check the initial (default) simulation of your model in SWAT-CUP do the following:

1-In Par_inf, put the number of simulations and the number of parameters to 1 and set up a dummy parameter change such as

r__CN2.mgt 0 0 (this does not change anything)

2-In SUFI2_swEdit put the beginning and the ending simulation also to 1

3-Then execute: Pre, Run, and Post processing.

Now look at the 95PPU result of your default or initial model run. If simulations and observations are too different, then you need to take a closer look at your swat model, including rainfall, etc.

If they are not too different, then for each outlet adjust relevant parameters in the relevant subbasins (referred to as parameterization), and do a few iterations based on that.

For a protocol see the open access paper:

http://www.sciencedirect.com/science/article/pii/S0022169415001985

How to set the parameters in SWAT text files to the best parameter values of the last iteration?

If you want the SWAT TxtInOut files reflect the best parameters you obtained in an iteration do the following:

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 both to the number of the best simulation in step 1.

3- Under Calibration, run SUFI2_run.bat. Do not run SUFI2-Pre.bat.

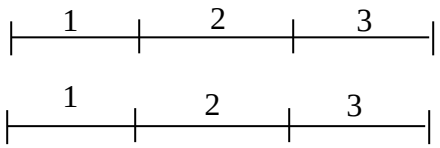
This command will replace the parameter values and set them to the best values of the last iteration.

How to do Latin Hypercube Sampling

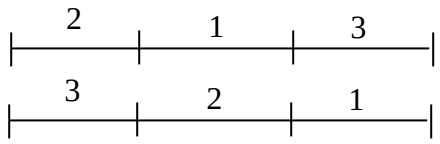
The batch file *SUFI2_pre.bat* runs the *SUFI2_LH_sample.exe* program, which generates Latin hypercube samples. These samples are stored in *par_val.txt* file.

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

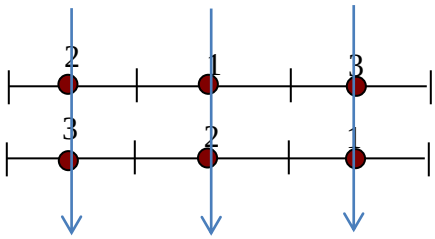
1) Parameters (say 2) are divided into the indicated number of simulations (say 3)



2) Parameter segments are randomized



3) A sample is taken at the middle of every segment



Every vertical combination is then a parameter set.

How to Calibrate more than one Variable

If you want to calibrate using, for example discharge, nitrate, and phosphate, you should first calibrate for discharge. This is because flow is the main controlling variable. After calibrating for flow, keep flow parameter ranges as you obtained from flow calibration and add sediment parameters. There are two types of sediment parameters, those that affect only sediment, and those that affect flow and sediment. See the Table below from Abbaspour et al., (2007).

Table 1 List of SWAT's parameters that were fitted and their final calibrated values

Variable	Sensitive parameters	Final parameter value
Parameters sensitive to all four variables	– snowfall temperature, SFTMP.bsn[a]	−1.1[b]
	– Melt factor for snow on December 21, SMFMN.bsn	0.36
	– Melt factor for snow on June 21, SMFMX.bsn	2.84
	– Snowmelt base temperature, SMTMP.bsn	2.8
	– Snowmelt temperature lag factor, TIMP.bsn	0.29
	– Baseflow alpha factor, v__ALPHA_BF.gw[c]	[0.17, 0.34]
	– Groundwater delay time, v__GW_DELAY.gw	0.74
	– Curve number, r__CN2.mgt	[0.085, 0.045]
	– Manning's n value for the main channel, v__CH_N2.rte	[0.0, 0.3]
	– Effective hyd. cond. in the main channel, v__CH_K2.rte	[4, 14]
	– Soil available water storage capacity, r__SOL_AWC.sol	[−0.17, 0.3]
	– Soil hydraulic conductivity, r__SOL_K.sol	[−0.19, 0.5]
	– Soil bulk density, r__SOL_BD.sol	[−0.02.7, 0.3]
	– Maximum canopy storage, v__CANMX.hru__AGRR[d]	2.8
	– Maximum canopy storage, v__CANMX.hru__FRST	4.8
	– Maximum canopy storage, v__CANMX.hru__PAST	4.1
Parameters sensitive to sediment only	– Sediment routing factor in main channels, v__PRF.bsn	[0.2, 0.25]
	– Channel re-entrained exponent parameter v__SPEXP.bsn	[1.35, 1.47]
	– Channel re-entrained linear parameter v__SPCON.bsn	[0.001, 0.002]
	– Channel erodability factor, v__CH_EROD.rte	[0.12, 0.14]
	– Channel cover factor, v__CH_COV.rte	[0.2, 0.25]
Parameters sensitive to total phosphorus only	– Phosphorus availability index, v__PSP.bsn	[0.5, 0.7]
	– P enrichment ratio with sediment loading, ERORGP.hru	[2.0, 4.0]
	– Rate constant for mineralization of organic P, BC4.swq	[0.3, 0.5]
	– Organic P settling rate, RS5.swq	[0.08, 0.1]
Parameters sensitive to nitrate only	– Nitrogen in rain, RCN.bsn	1.3
	– Nitrogen uptake distribution parameter, UBN.bsn	9.4
	– Concentration of NO_3 in groundwater, r__GWNO3.gw	[−0.3, 0.5]
	– Organic N enrichment for sediment, ERORGN.hru	2.75
	– Nitrate percolation coefficient, NPERCO.bsn	0.223
Parameters sensitive to sediment and total phosphorus	– support practice factor r__USLE_P.mgt	[−0.6, −0.1]
	– water erosion factor v__USLE_C.crp____AGRR	[0.03, 0.3]
	– water erosion factor v__USLE_C.crp____PAST,ORCD	[0.07, 0.2]
	– water erosion factor v__USLE_C.crp____FRST	[0.0, 0.1]
	– soil erodability factor, r__USLE_K.sol	[−0.19, 0.5]

[a] The extension (.bsn) refers to the SWAT file type where the parameter occurs.
[b] The fixed values indicate that a parameter was fitted and then fixed.
[c] The qualifier (v__) refers to the substitution of a parameter by a value from the given range, while (r__) refers to a relative change in the parameter were the current values is multiplied by 1 plus a factor in the given range.
[d] AGRR = agricultural, PAST = pasture, ORCD = orchard, FRST = forest.

Initially, add the parameters that only affect sediment and run an iteration. You should get the same discharge results as before, so try calibrating only for the sediment parameters that don't affect the flow

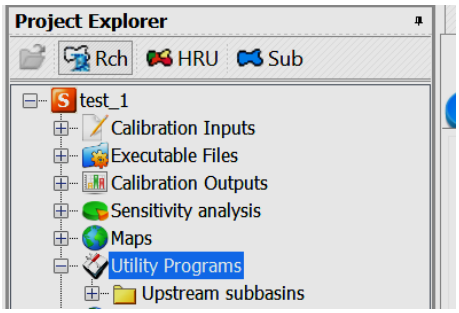
first. After, one or two iterations, if sediment results are not satisfactory, then add the other parameters that affect sediment and flow and do a couple of iterations by allowing flow parameters to also change slightly.

For Nitrate repeat the same procedure with nitrate parameters. Note that for calibrating phosphor you must calibrate for sediment first, because much of the phosphor moves with sediment, but nitrate can be calibrated without sediment.

It is important to also note that the solution to the calibrated model is the 95PPU generated by the parameter ranges. DO NOT try to only use the best parameter set for further analysis. By doing this you are assuming that the calibrated model only has one solution and this is not correct. It is never correct to assume that only one set of parameters can represent a watershed, which was modeled by very uncertain information about soil, landuse, climate, management, measured data used for calibration, etc. Always propagate the range of parameters you obtained during calibration for all purposes of model use.

Utility programs

(……..\ExternalData\utilityprograms)



This module currently has two program in it: Make_ELEV_BAND, not shown in the interface, and Upstream subbasins, which is shown in the interface.

Make_ELEV_BAND

This program can calculate the elevation band for a SWAT project and use SWAT-CUP to put the information in SWAT *.sub files. There is an explanatory file called elev_band.doc, which explains how to do this.

Upstream subbasins

This program can determine the upstream subbasins. This is a useful information for parameterization. An explanation is given as follows.

Upstream_subbasins.exe

This program finds all the upstream subbasin for given outlets. We use the following example to illustrate.

input:

1- upstream.def

- In this file you specify total subbasins in the project,
- the number of subbasins (or outlets) you want to know their upstream subbasins,
- and the list the outlet numbers

```
20        : number of subbasins
4         : number of outlets

1
3
7
18
```

2- upstream_from_to.txt

- In this file you copy is from columns “from” and “to” from the SWAT projects .mdb database.
- Open project's .mdb file --> Reach --> copy the columns 'from' and 'to' into this text file

FROM_NODE	TO_NODE
1	4
2	0
4	7
5	3
6	3
7	3
8	7
9	7
11	10
12	10
13	10
14	13
16	15
17	15
18	13
19	18
3	2
10	8
15	13
20	18

output:

1- upstream.out

This file contains upstream subbasin numbers of the given outlets as well as the number of outlets, i.e., outlet 1 has only 1 upstream subbasin, outlet 3 has 19 upstream subbasins, etc.

```
outlet_no=   1, 1
1,

outlet_no=   3, 19
3,1,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,

outlet_no=   7, 16
7,1,4,8,9,10,11,12,13,14,15,16,17,18,19,20,

outlet_no=   18, 3
18,19,20,
```

subtract_subbasins.exe

This program subtracts common subbasins away from the two given lists of upstream subbasins for two given outlets. There is one input and one output file.

input:

subtract.def

Copy two desired outlets from upstream.out into this file.

```
outlet_no=   3, 19
3,1,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,

outlet_no=   7, 16
7,1,4,8,9,10,11,12,13,14,15,16,17,18,19,20,
```

Output:

subtract.out

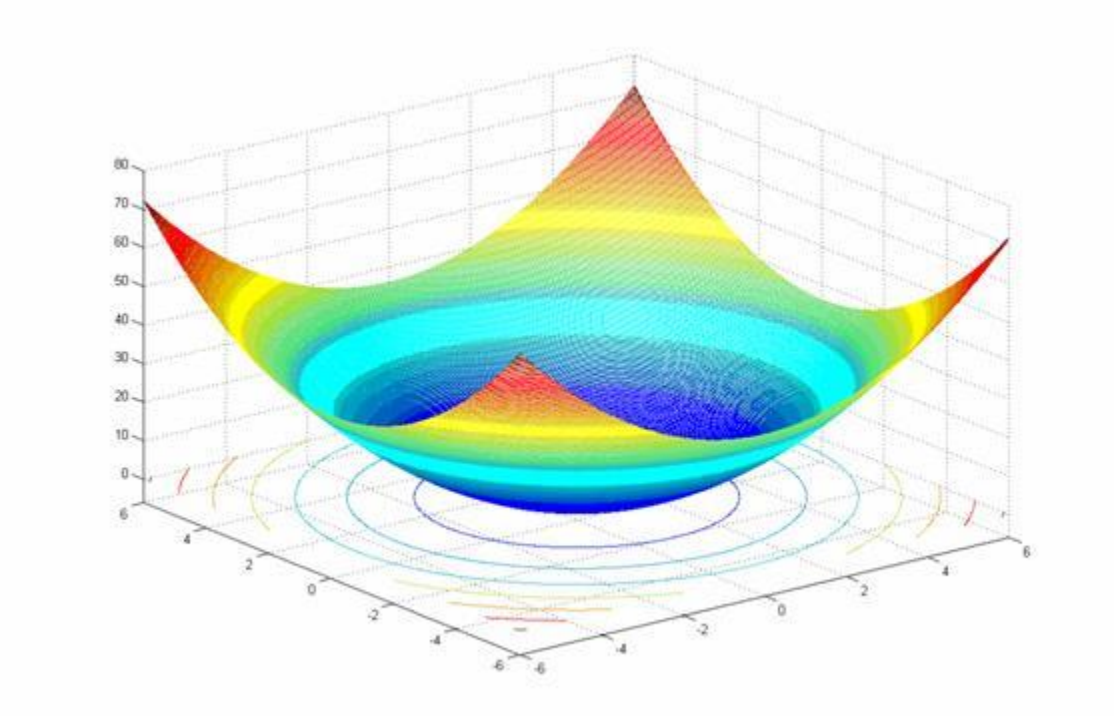
Run the subtract_subbasins.exe to get the output in subtract.out file

```
4,8,9,1,10,11,12,13,
counter= 8
```

this file has the list of subbasins that are not common between the two subbasins.
Therefore, to parameterize for outlet number 18, you would use parameters of subbasins 18, 19, and 20.
To parameterize for outlet number 7, you would use subbasins 7,1,4,8,9,10,11,12,13,14,15,16, and 17.
And to parameterize for subbasin number 3, you would use subbasins 3,5, and 6.

PSO

Particle Swarm Optimization



Introduction to PSO

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.

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)

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 the following equations (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.

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.

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.

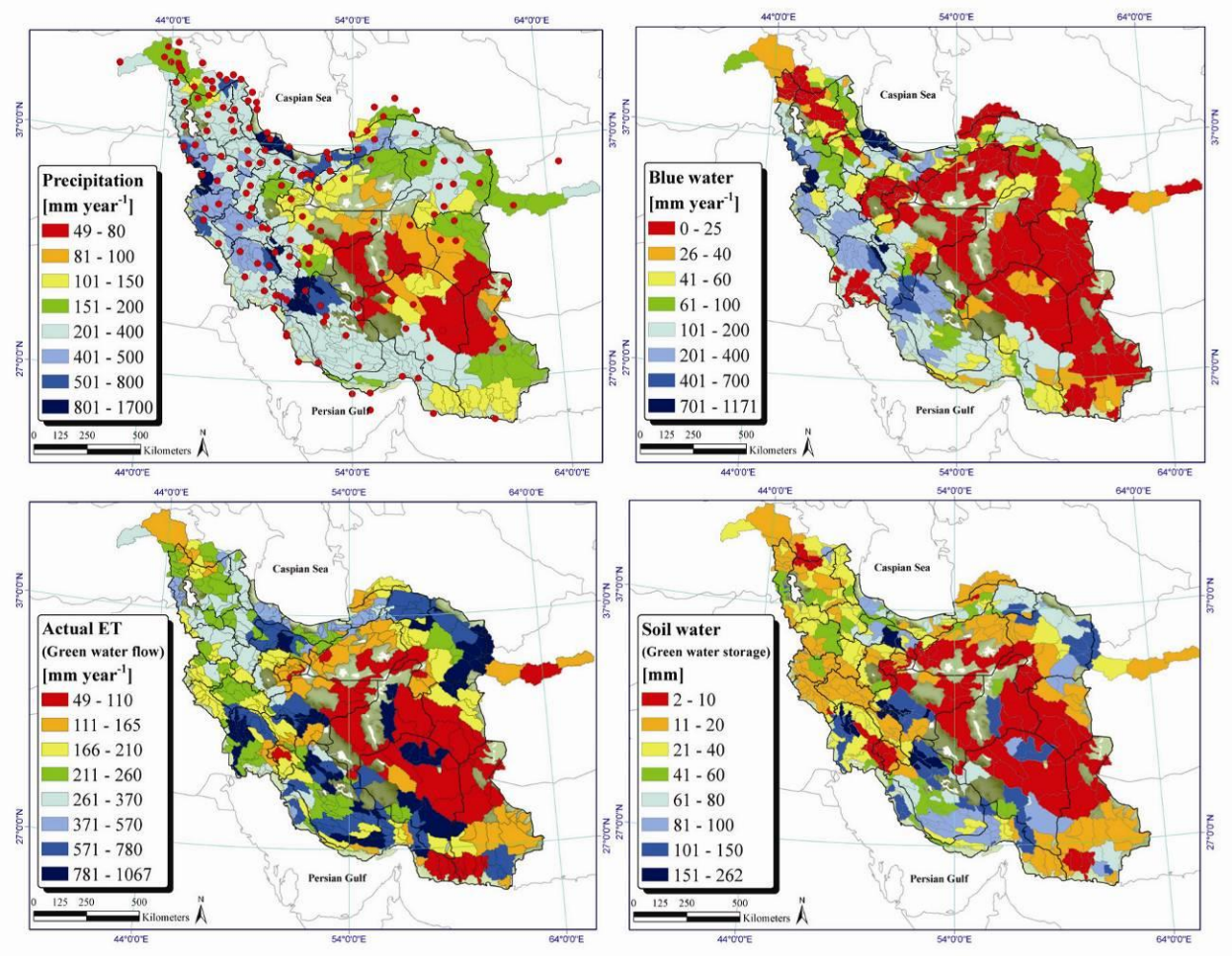
Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization. Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf., Springer-Verlag, Berlin, San Diego, CA., 1998.

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

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:

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

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.

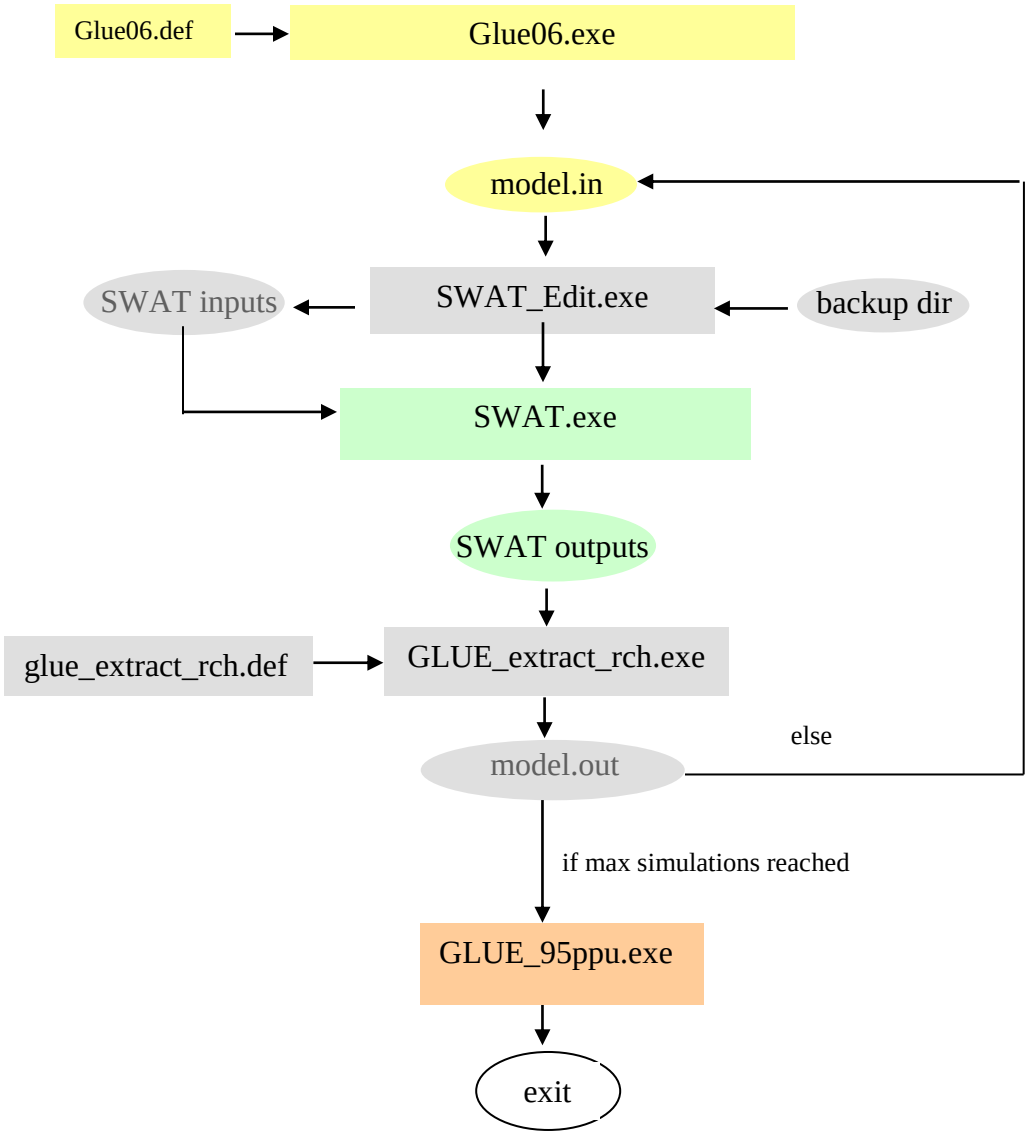
In literature, the most frequently used likelihood measure for GLUE is the *Nash-Sutcliffe* coefficient (*NS*), which is also used in the GLUE06 program:

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

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 $\overline{y}$ is the average value of the observations.

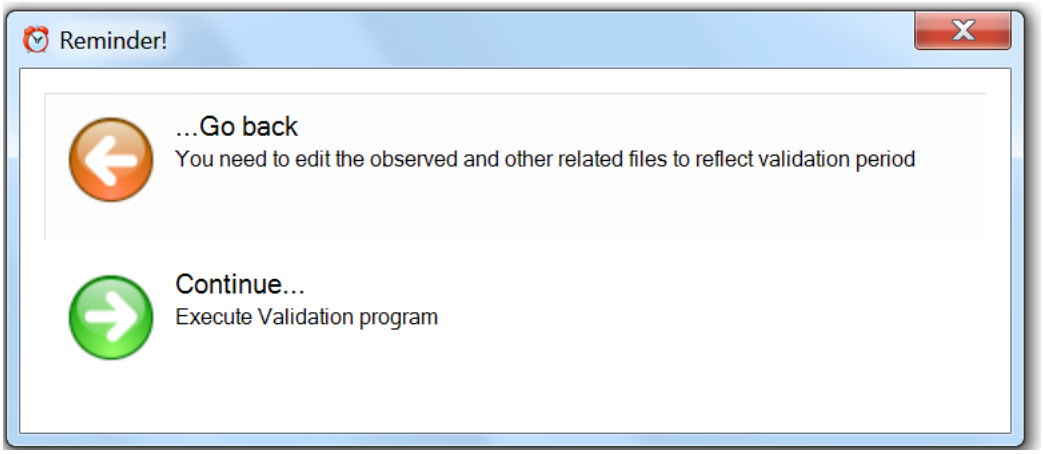
Coupling of GLUE to SWAT-CUP

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. Interface of GLUE and SWAT-CUP is as follows:



Validation of GLUE

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 as indicated in the window that appears when validation is executed. The validation program uses the behavioral parameters only to run SWAT.



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

File Definition

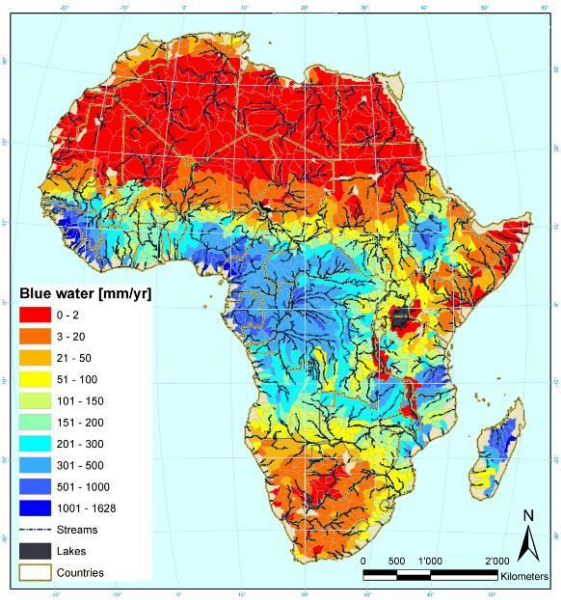
glue06.def

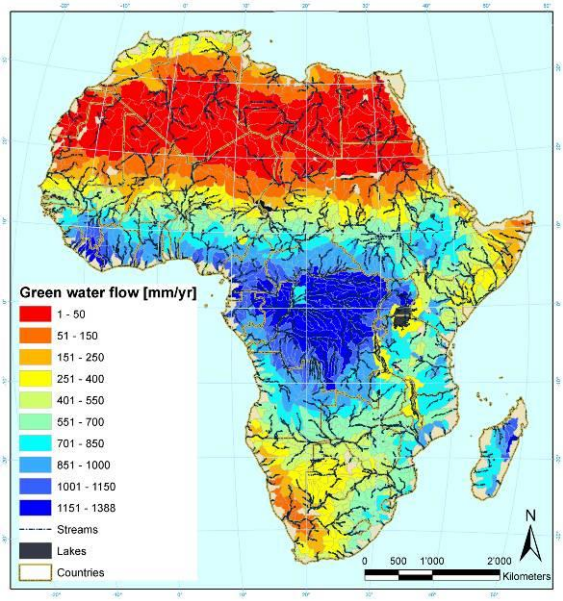
Line	parameter		value	Remark
1	// comment			
2	MaxSimulation		10000	The larger, the better!
3	ParDefFile		glue_par.def	parameter definition file
4	ObjfunThresh		0.3	Threshold value given by the user to separate the behavioural parameters from the non-behavioural parameters
5	Percentile		0.025	The percentile 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

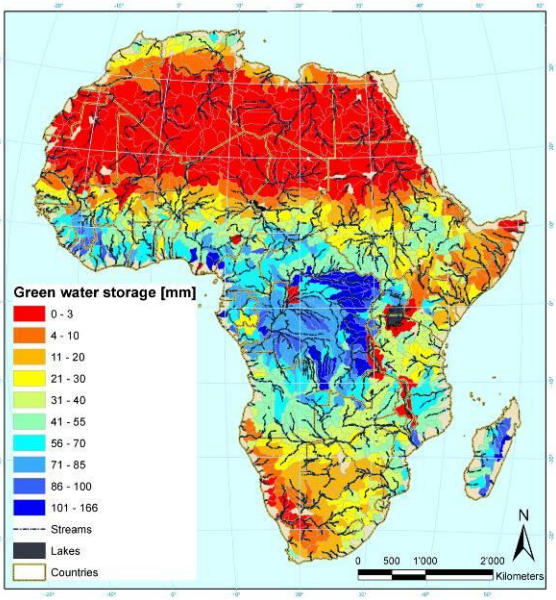
11	ModelBehParaset		modelpara.beh	The output filename for the behavioural parameter sets
12	ModelResult	T	modelres.out	The output filename for all the model results
13	ModelBehResult		modelres.beh	The output filename for the behavioural model results
14	ModelResQaunt	T	modelquant.out	The output filename 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.

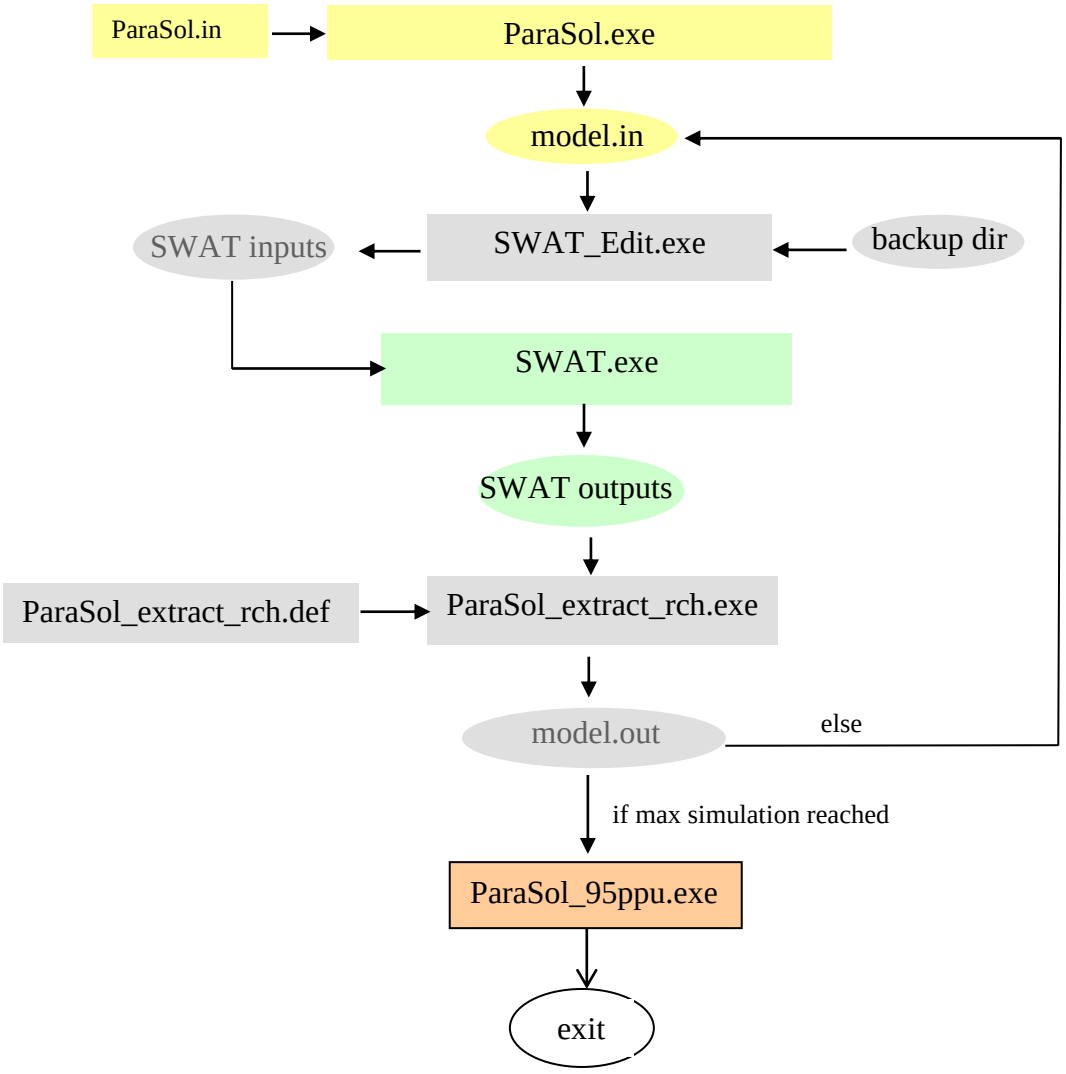
2) The prediction uncertainty is hence constructed equally from the 'good' simulations.

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.



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

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.

ParaSol: optimization and uncertainty analysis tool

Ann van Griensven and Tom Meixner

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)

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

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.

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

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

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

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

$$\sigma^2 = \frac{SSQ_{MIN}}{nobs} \quad (3)$$

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:

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

or

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

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

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

or

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

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

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

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:

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

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

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

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

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

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:

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

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 (θ^*_1, θ^*_2,… θ^*_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

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

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:

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

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

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

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

$$PT = \sum_{i=1}^{T} W(\theta_i) * p(\theta_1 \mid Y_{obs}) \quad (17)$$

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.

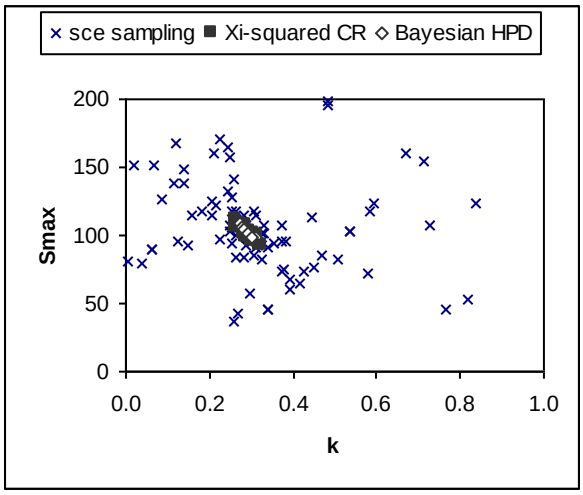


Figure 2: Confidence region CR for the χ^2-statistics and high probability density (HPD) region for the Bayesian statistics for a 2-parameter test model

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.

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.

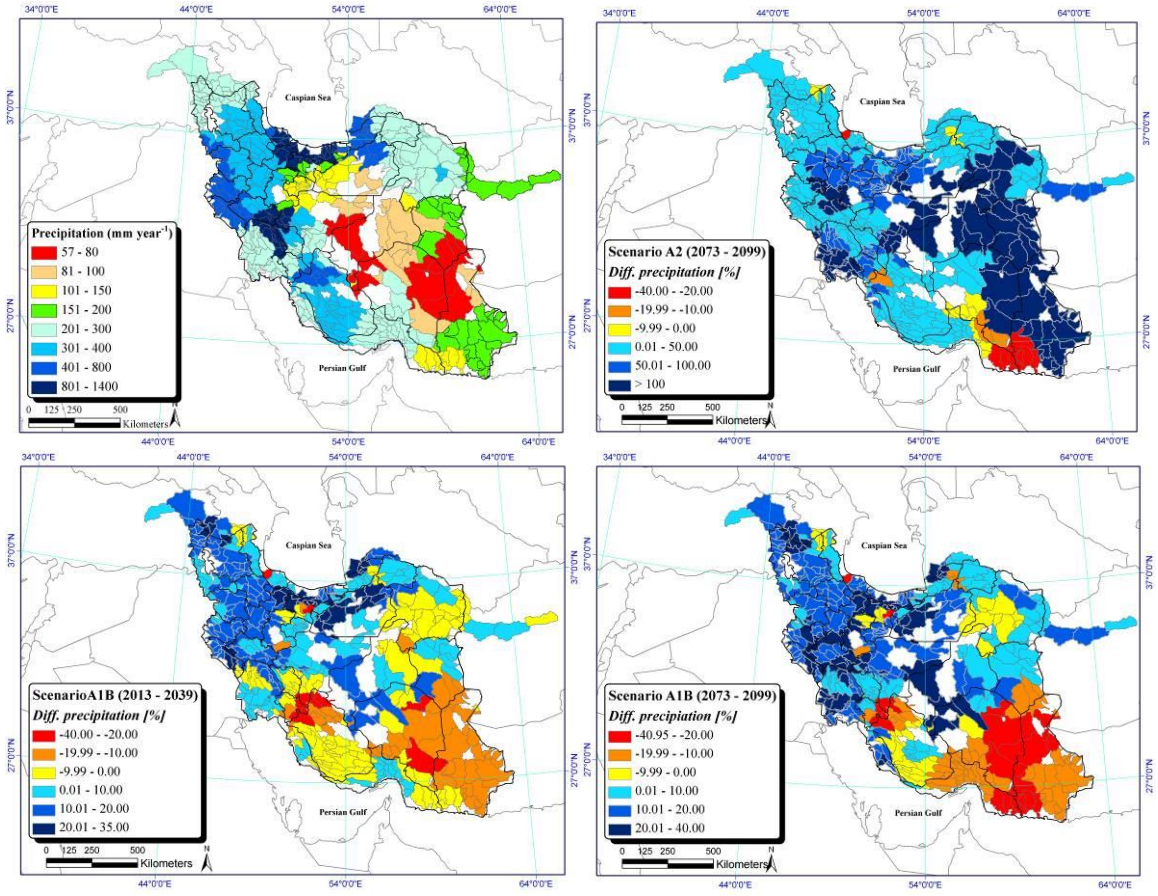
Output files

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.

Markov Chain Monte Carlo



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*:

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

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

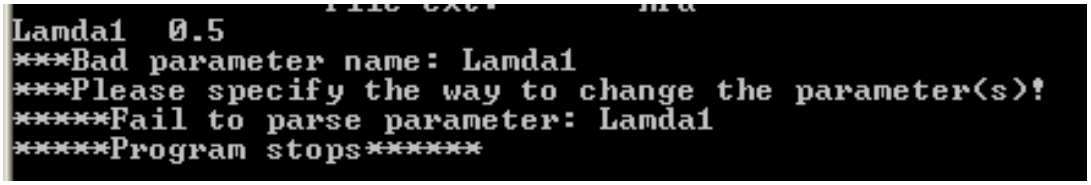
The jump distributions are quite important to convergence and require some initial trial and error runs to specify.

mcmc_run.bat

SWAT_Edit.exe	//program to insert generated parameters in swat input files
swat2005.exe	//swat program either swat2000 or swat2005
MCMC_extract_rch.exe	//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:



Reference

Abbaspour, K. C. E. Rouholahnejad, S. Vaghefi, R. Srinivasan, B. Klöve. **2014**. Modelling hydrology and water quality of the European Continent at a subbasin scale: calibration of a high-resolution large-scale SWAT model. **Journal of Hydrology**, 524: 733-752.
http://www.sciencedirect.com/science/article/pii/S0022169415001985

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, 1998. Toward improved calibration of hydrologic models: multiple and noncommensurable measures of information, *Water. Resourc. Res., 34*:751-763.

Gupta, H. V., S. Sorooshian, and P. O. Yapo. 1999. Status of auto-matic calibration for hydrologic models: Comparison with mul-tilevel expert calibration. *J. Hydrologic Eng.*, 4(2): 135-143

Gupta, H.V., Kling, H., Yilmaz, K.K., Martinez, G.F.2009. Decomposition of the mean squared error and NSE performance criteria: implications for improving hydrological modelling.J.Hydrol.377, 80–91.

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.

Krause, P., D.P. Boyle, F. Bäse, 2005. COmparison of different efficiency criteria for hydrological model assessment, Adv. In Geoscheices, 5:89-97.

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, 1999. Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. Water. Resou. Res., 35:233-241.

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.

Moriasi, D.N., Arnold, J.G., Van Liew, M.W., Bingner, R.L., Haemel, R.D., Veith, T.L. 2007. Model evaluation guidelines for systematic qualification of accuracy in watershed simulation. Transactions of the ASABE, 50:885-900.

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, Abbaspour KC, Vejdani M, Srinivasan R, Schulin R, Lehmann A. 2012. Parallelization framework for calibration of hydrological models, *Environmental Modelling Software,* 31: 28-36.

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.