ERDC/CHL TR-12-2

Coastal and Hydraulics Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

A Practical Guide to Calibration of a GSSHA Hydrologic Model Using ERDC Automated Model Calibration Software – Effective and Efficient Stochastic Global Optimization

Brian E. Skahill, Charles W. Downer, and Jeffrey S. Bagget

February 2012

ERDC/CHL TR-12-2
February 2012

A Practical Guide to Calibration of a GSSHA Hydrologic Model Using ERDC Automated Model Calibration Software – Effective and Efficient Stochastic Global Optimization

Brian E. Skahill and Charles W. Downer

Coastal and Hydraulics Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS39180

Jeffrey S. Baggett

University of Wisconsin at La C rosse
1725 State Street
La Cross,WI 54601

Final report

Prepared for U.S. Army Corps of Engineers
441 G. Street, NW
Washington, DC 20314-1000

Abstract

The objective of this article is to demonstrate, by way of example(s), how to use our implementation of the MLSL method for model independent parameter estimation to calibrate a GSSHA hydrologic model. The purpose is not to present or focus on the theory which underlies the parameter estimation method, but rather to carefully describe how to use the ERDC software implementation of MLSL that accommodates the PEST model independent interface to calibrate a GSSHA hydrologic model. Given the computational expense associated with global optimization, we will initially consider variations of our MLSL implementation on a computationally efficient test problem in attempts to provide the interested reader with an intuitive sense of how the method works.

Contents

Figures and Tables

Preface

This report describes, by way of multiple examples, how to use model independent software to calibrate the physics based, distributed parameter, hydrologic model Gridded Surface Subsurface Hydrologic Analysis (GSSHA).

Research presented in this technical report was developed under the U.S. Army Corps of Engineers Flood and Coastal Storm Damage Reduction Research and Development Program. Dr. William Curtis, Coastal and Hydraulics Laboratory (CHL), is the director.

The work was performed by Drs. Brian E. Skahill and Charles W. Downer of the Hydrologic Systems Branch (HF-H) of the Flood and Storm Protection Division (HF), U.S. Army Engineer Research and Development Center – Coastal and Hydraulics Laboratory, and Dr. Jeffrey S. Baggett of the University of Wisconsin – La Crosse. At the time of publication, Earl V. Edris was Chief, (HF-H); Bruce A. Ebersole was Chief. The Deputy Director of CHL was Jose E. Sanchez and the Director was Dr. William D. Martin.

COL Kevin J. Wilson was the Commander and Executive Director of ERDC, and Dr. Jeffery P. Holland was the Director.

1 Introduction

Recent research at the U.S. Army Engineer Research and Development Center (ERDC) has focused on the development of methodologies or improvement of the efficiency of native algorithms, for the computer-based calibration of hydrologic and environmental models (wherein by efficiency, we mean the number of forward model calls necessary for the calibration algorithm to converge). These include, among others, an accelerated derivative-based local search algorithm, a stochastic global optimization algorithm for intelligently sifting through local minima to find a global minimum and, most recently, a state-of-the-art evolutionary strategy for global parameter identification of difficult problems with noise or other features that make derivatives estimation difficult. Minimizing the number of required model runs is one of the primary factors driving the research and development activities, such that the resulting optimization tool(s) are more compatible with the computationally expensive physics-based models that are becoming more commonly used within the practice community.

In a previous technical report (Skahill et al. 2011) we recently discussed how to use, by way of example, our implementation of the Levenberg-Marquardt (LM) local search method (Levenburg, 1944; Marquardt, 1963), and also the Secant LM (SLM) method, an efficiency enhancement to the LM method, for computer-based model independent hydrologic model calibration. The context for this article will be directed to the previously mentioned stochastic global optimization algorithm, which at present uses our implementations of the LM/SLM methods for local search. The LM method has features that make it attractive for model calibration. One feature is its ability to readily report estimates of parameter uncertainty, correlation, and (in)sensitivity as a by-product of its use both during and after the parameter estimation process. Another feature is that it is easily adapted by the inclusion of various regularization devices to maintain numerical stability and robustness in the face of potential numerical problems (that adversely affect all parameter estimation methodologies) caused by parameter insensitivity and/or parameter correlation (Menke, 1984; de Groot-Hedlin and Constable, 1990; Doherty and Skahill, 2006). Skahill et al. (2009) and Skahill and Doherty (2006) both provide lengthy summaries of the LM method.

The model independent LM method based parameter estimation software PEST (Doherty, 2004, 2007a, 2007b), which quantifies model to measurement misfit in the weighted least squares sense, is now widely used to support hydrologic and environmental model calibration. In addition to its traditional groundwater model calibration application setting (Zyvoloski et al., 2003; Tonkin and Doherty, 2005; Moore and Doherty, 2006; Gallagher and Doherty, 2007a), it is now employed to calibrate ecological models (Rose et al., 2007), land surface models (Santanello Jr. et al., 2007) and models in other application areas including nonpoint source pollution (Baginska et al., 2003; Haydon and Deletic, 2007), surface hydrology (Doherty and Johnston, 2003; Gutiérrez-Magness and McCuen, 2005; Kunstmann et al., 2006; Skahill and Doherty, 2006; Doherty and Skahill, 2006; Gallagher and Doherty, 2007b; Goegebeur and Pauwels, 2007; Iskra and Droste, 2007; Kim et al., 2007; Maneta et al., 2007), and surface water quality (Rode et al., 2007).

Skahill et al. (2011) focused on one drawback associated with LM-based model independent parameter estimation as implemented in PEST; viz., that it requires estimates, based on finite differences, of the derivatives of the objective function with respect to the model parameters. We briefly discussed the secant LM method and then presented examples which demonstrated benefits that can be derived in terms of improved inverse model run-efficiency when using the SLM method rather than LM for calibrating a GSSHA hydrologic model deployment for the Goodwin Creek Experimental Watershed (GCEW).

However, as mentioned, another drawback of the LM method is that it is a local search method. Thus, if there are different "regions of attraction" in parameter space, its solution will lead to just one of possibly many objective function minima, the particular one that is found is dependent upon the user-supplied set of initial parameter values. Stochastic global optimization (GO) can be employed as a remedy. Stochastic global optimization algorithms estimate the global minimum of the objective function by initiating local searches from global, randomly sampled points. The local and global phases can be iterated and/or the local searches may be initiated at some or all of the globally sampled points. Stochastic global optimization algorithms are guaranteed to converge, with probability one, to the global minimum as the sample size approaches infinity. Stronger convergence properties are possible for some stochastic algorithms, as we mention below. Moreover, probabilistic-based stopping criteria can be developed for stochastic global optimization methods (Rinnooy Kan and Timmer, 1987a, b; Törn and Žilinskas, 1987); however, an a priori computational budget may preclude any concern regarding termination criteria.

One would like to utilize stochastic GO methods that are not only reliable in finding the global minimum, but also efficient in the sense that they minimize the return to previously visited local minima in parameter space. A modeler would possibly also like to receive some information on the locations of non-global minima, especially if these minima are little different in magnitude from the global minimum, but are widely separate from it in parameter space. With the understanding that efficient and reliable optimization methods, possibly constrained by a predetermined computational budget, that are capable of efficiently finding the locations of other good minima in addition to an estimate of the global minimum, are needed to identify hydrologic and/or environmental models, Skahill et al. (2009) implemented a more efficient and reliable stochastic global optimization algorithm than what is currently available in PEST.

Ideally, we would like to perform a single local search within the region of attraction of each local minimum. This would not only ensure that each local minimum is identified just once, but also that in fact we find all local minima. But we also want to employ a method that works well if one has a predetermined computational budget in that for a given effort it compares favorably with other methods. Clustering methods were designed to accommodate these requirements. They are variants of Multistart (the Multistart method samples points from a uniform distribution over the feasible parameter space and starts a local search from each of the sample points) and the basic idea behind them is to group close points, sampled from the feasible parameter space and for which the specified groups presumably relate to actual regions of attraction in parameter space, and to apply a single local search procedure within each identified cluster. Either reduction; wherein sampled points associated with the highest objective function values are temporarily removed, or concentration; wherein the sampled points are transformed through application of one or a few iterations of a local search procedure, is employed to identify a reduced sample as part of the clustering process in order to provide some assurance that in fact the specified groups correspond to regions of attraction of actual local minima. Clustering methods are often iterative in that the global and local phases are repeated sequentially until a stopping criterion is satisfied.

With clustering methods, it is possible that one cluster intersects multiple regions of attraction; hence, the global minimum could be missed, or that one region of attraction contains more than one cluster, thus allowing for the same local minimum to possibly be identified more than once. Multi Level Single Linkage (MLSL) is a clustering method that was developed to reduce the probability of not finding a local minimum or of finding a local minimum more than once (Rinnooy Kan and Timmer, 1987a, b).

MLSL mimics clustering by calculating a critical distance *rk* at each iteration, *k*. This critical distance can be used to build clusters, but instead, in MLSL, the decision as to whether a local search is to be initiated from a given reduced sample point is simply based on whether there exists another reduced sample point within the distance *rk* of the given point with a corresponding lower objective function value. The critical distance *rk* is reduced at each iteration.

Under certain assumptions, MLSL has stronger convergence properties than simpler stochastic global optimization algorithms. First, if the algorithm continues forever, the number of local searches performed is finite. Second, if *rk* tends to zero with increasing *k*, then every local minimum will be identified in finite time with probability one.

We implemented MLSL and it uses the ERDC implementation of the LM or SLM method for local search. Our MLSL implementation follows that of Rinnooy Kan and Timmer (1987a, b) with a slight modification to sometimes avoid repeatedly finding the same local minima. Our MLSL implementation is summarized in Figure 1. The following stopping criteria were included as part of our MLSL implementation:

1. The objective function has not been lowered over a specified number of local searches,
2. A specified maximum number of local searches have been performed,
3. The expected number of minima, in the Bayesian sense, exceeds the number of identified distinct local minima by less than 0.5 (Rinnooy Kan and Timmer, 1987a, b),
4. A specified maximum number of MLSL iterations have been performed,
5. The objective function has not been lowered over a specified number of MLSL iterations.

Objective. $\min_{x \in X} f(x)$, $X \subseteq \Re^d$.

Initialization. Let S^* denote the set of identified local minima. Start with $S^* = \emptyset$. Let the integer $N > 0$, $\gamma \in (0,1]$ and d_1, $d_2 > 0$ be fixed, let $k := 1$ and

$$r_k := \pi^{-1/2}\left(\Gamma\left(1+\frac{d}{2}\right)m(X)\sigma\frac{\log kN}{kN}\right)^{1/d}$$

Sampling phase. Use a uniform distribution to sample N points from X at step k.

Reduction of the sample. Sort the entire sample of kN points in order of increasing objective function values. Select the γkN points with the lowest objective function values. This resultant set, R_k, is called the reduced sample.

Selection of the set S_k. Apply a local search to every point x_i in the reduced sample set which satisfies the following:

1. There is no other point, y, in R_k or S^* such that $d(x_i, y) \leq r_k$ and $f(y) < f(x_i)$
2. $d(x_i, S^*) > d_2$
3. A local search has not already been applied to x_i.

Stopping phase. If the termination criteria are not satisfied, then return to the sampling phase with $k = k + 1$.

Figure 1. Our implementation of the Multi Level Single Linkage (MLSL) algorithm.

An instance of our MLSL implementation stops when any one of the above criteria are satisfied. It can be utilized by simply appending the MLSL search parameters to the end of the control data section of a working *SLM_CHL* input control file (see Skahill et al. 2011). Hence, as with our independent LM/SLM implementations, only slight modification to a working *PEST* input control file is required to also use our MLSL implementation.

Skahill et al. (2009) performed an efficiency comparison of our MLSL implementation with Trajectory Repulsion (Skahill and Doherty, 2006), the

Shuffled Complex Evolution (SCE) general purpose global optimization method (Duan et al. 1992; Duan et al. 1993) and the Covariance Matrix Adaption Evolutionary Strategy (CMAES) general purpose optimization method (Hansen and Ostermeier, 2001; Hansen et al. 2003), using an eight parameter Hydrological Simulation Program-FORTRAN (HSPF) (Bicknell et al., 2001) hydrologic model, a ten parameter Fast All-season Soil Strength (FASST) state-of-the-ground model (Frankenstein and Koenig, 2004), and a sixteen parameter Gridded Surface Subsurface Hydrologic Analysis (GSSHA) (Downer and Ogden, 2003a, 2003b) hydrologic model. In particular, using the three previously mentioned environmental models, Skahill et al. (2009) compared the efficiencies, in terms of the number of model calls required to achieve a given objective function value, of their implementations of Trajectory Repulsion and MLSL with that of SCE and CMAES as implemented in PEST (Doherty, 2004, 2007a, 2007b). To examine their relative efficiencies, thirty trials of MLSL, Trajectory Repulsion, CMAES, and SCE were conducted with each of the three distinct environmental model structures. While PEST includes a Trajectory Repulsion implementation, our Trajectory Repulsion implementation was utilized so that the local search implementation (SLM) would be identical to that employed with MLSL.

Results from the numerical experiments were obtained by computing the average over thirty trials of the best objective function value obtained after a specified number of model calls. Skahill et al. (2009) pointed out that they were very wary of comparing different software packages in this manner, for a package, and the methodology which it encapsulates, always performs best when operated by its designers. This is because program settings, particularly those pertaining to termination and convergence criteria, can have a huge effect on the performance of a method; a non-expert in the use of a particular package may not be aware of the optimal settings to use, especially in difficult cases. Hence, we do not pretend that their results provide a comprehensive basis for assessment of the comparative performance of *SCE-PEST*, *CMAES-PEST* and *MLSL*. We hope, however, that they do provide a basis for at least a “ball park” comparison of the methods for the particular calibration cases they considered.

Based on the numerical experiments involving thirty trials with each global optimization method, and for each of the three model structures, Skahill et al. (2009) recommend *MLSL* over Trajectory Repulsion for environmental model independent LM-based stochastic global optimization. Moreover, their results also suggested potential utilization of MLSL over SCE and CMAES, except for the case where only a very limited computational budget is available, in which case CMAES might be preferable.

The objective of this article is to demonstrate, by way of example(s), how to use our implementation of the MLSL method for model independent parameter estimation to calibrate a GSSHA hydrologic model. The purpose is not to present or focus on the theory which underlies the parameter estimation method, but rather to carefully describe how to use the ERDC software implementation of MLSL that accommodates the PEST model independent interface to calibrate a GSSHA hydrologic model. Given the computational expense associated with global optimization, we will initially consider variations of our MLSL implementation on a computationally efficient test problem in attempts to provide the interested reader with an intuitive sense of how the method works.

2 Examples

The steps necessary to use our implementation of the MLSL GO method will now be demonstrated and documented while applying it to initially calibrate a computationally efficient test problem and subsequently a GSSHA hydrologic model for the Goodwin Creek Experimental Watershed (GCEW) (Senarath et al. 2000; Downer and Ogden, 2003b). The general approach simply involves slight modification of an existing functional input control file associated with the ERDC implementation of LM/SLM. The interested reader is referred to our recent report for details on how to interface a given forward model with the ERDC LM/SLM implementations (Skahill, Downer, and Baggett, 2011).

Example 1

In this first example, we will document the steps necessary to use the ERDC implementation of the MLSL stochastic global optimization method to calibrate the computationally efficient test problem defined directly below in Figure 2.

Problem Find p_1 and p_2 to minimize:

$$\sum_{i=0}^{86}\Big((\sin(p_1 * x_i) + \sin(p_2 * x_i)) - (\sin(\frac{2}{3} * x_i) + \sin(x_i))\Big)^2$$

where $x_i = 3.1 + 0.15i$ and $p_1 \in [0, 2], p_2 \in [0, 2]$. The global min is at $p_1 = \frac{2}{3}$, $p_2 = 1$ (or vice versa by symmetry). This problem has a bunch of local minima. This makes it challenging because no method will find either of the global minima easily.

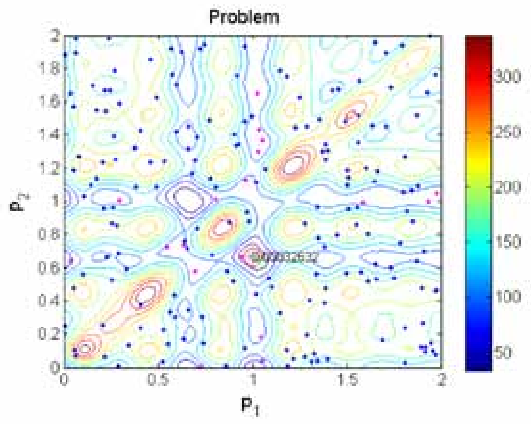


Figure 2. Implementation of MLSL with many local minima.

Step 01 – Obtain forward model

The *C* source code provided directly below, once built (it was named *tp2.exe*), computes model generated observations for a given parameter set ((p_1,p_2)). The program reads in a file named "*tp2_parms.txt*", a file with two floating point data values, one specified on the first row, and the other specified on the second row of the file, respectively. The first row entry is the value for the first parameter, p_1; whereas, the second row entry is the value for the second parameter, p_2. The 87 model computed observations are generated and written to a file named "*tp2_out.txt*", with one floating point data value specified per line.

```
////////////////////////////////////////////////////
//Example Test Problem for testing MLSL code
//
//Author: Brian E. Skahill
//
///////////////////////////////////////////////////

#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
#include "string.h"
#include "ctype.h"
#include "math.h"
#include "time.h"

void main( void )
{
	int i;
	double h[87], value, m[3], x=0.0;
	FILE *data, *read;

	data = fopen( "tp2_out.txt", "w" );
	read = fopen( "tp2_parms.txt", "r" );

	////////////////////////////////////////////////////
	//Read the parameter values - m
	////////////////////////////////////////////////////
	for( i = 1; i <= 2; i++ )
	{
		value = 0.0;
		fscanf( read, "%lf\n", &value );
		m[i] = value;
	}
```

```
        ////////////////////////////////////////////////////
        //Compute h
        ////////////////////////////////////////////////////
        for( i = 0; i <= 86; i++ )
        {
                x = 3.1 + 0.15*i;
                h[i] = (sin(m[1]*x)+sin(m[2]*x));
        }

        ////////////////////////////////////////////////////
        //Write the computed h to file
        ////////////////////////////////////////////////////
        for( i = 0; i <= 86; i++ )
                fprintf( data, "%16.14E\n", h[i] );

        fclose( data );
}
```

Step 02 – Collect observed data

Observations for the test problem can be generated by running the model executable for the problem, tp2.exe, whose source code was presented and briefly described in the previous step, with input parameter values set to $p_1 = 1$ and $p_2 = 2/3$, or vice versa. Upon execution of *tp2.exe* at $p_1 = 1$ and $p_2 = 2/3$ (viz., the global minimum), the (synthetic) observations for the test problem will be located in the file named "*tp2_out.txt*", as mentioned above. The observations for the test problem will be discussed further below and clearly noted at that time.

Step 03 – Prepare instruction file for forward model output file

As with the ERDC LM/SLM implementations, our MLSL implementation is model independent. Hence, there is a need with each forward model call during an inverse model run to be able to read from one or more model generated output files the model generated observations relevant to objective function evaluation. An instruction file is the means to satisfy this requirement. As was also mentioned in our recent technical report, our software was written to accommodate the popular PEST model independent and input control file protocol; hence, the interested reader is directed to Doherty (2004, 2007a, b, c) for additional details regarding the PEST model independent interface, and in particular, instruction file development. The model instruction file for the test problem, named "*tp2_out.ins*", is provided in Appendix 1 (Please note that all of the appendices associated with this technical report are available in a separate document made publicly available on the GSSHA Knowledge Hub at https://knowledge.usace.army.mil/). It is the basis for reading model generated observations from the model (*tp2.exe*) output file named "*tp2_out.txt*".

Step 04 – Prepare template file for forward model input file

For this example, the model input file named "tp2_parms.txt" contains the two input parameter values that are designated as adjustable. During an inverse model run (i.e., model calibration), this model input file will need to be updated every time there is a need to evaluate the model at a new location in adjustable model parameter space. To support the interface of the model for this test problem with the independent PEST and ERDC LM/SLM implementations, a PEST template file was prepared for this model input file. The contents of the template file named "*tp2_parms.tpl*" are shown directly below.

```
ptf $
$p1 $
$p2 $
```

Step 05 – Prepare preliminary input control file

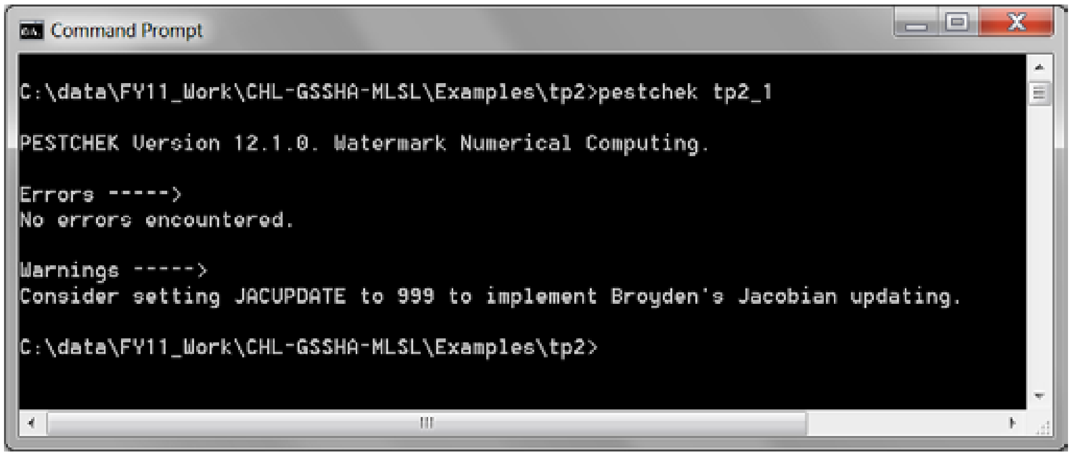
We are now ready to generate an input control file, the main input file for execution of the LM method not only associated with the PEST, but also our own independent LM/SLM implementation. The path that we take for this two parameter test problem will be different from the path that we pursued to generate an input control file for the GSSHA GCEW model, as we documented in our recent technical report (Skahill et al. 2011). In this case, we will simply take an existing working control file and manually modify it in a text editor so that it is specific to our two parameter test problem being considered in this example. In particular, we will manually modify the control file listed in Appendix 10 of our recent technical report (Skahill et al. 2011). Relevant changes to the control file will be mentioned so that the reader can interpret and follow the necessary changes. The final working input control file for this example problem, named *tp2_1.pst*, is listed in Appendix 2. As was mentioned in our recent technical report, the interested reader is referred to the PEST documentation for explanations related to control files. The second row of the control data section of the control file is changed from "16 233 16 0 1" to "2 87 2 0 1" because we now have two adjustable model parameters, two parameter groups, and 87 observations rather than 16 adjustable model parameters, 16 parameter groups, and 233 observations. Interpreting the parameter groups and parameter data sections of the input control file is relatively straight forward, and the interested reader is directed to the PEST documentation and also our recent report (Skahill et al. 2011) for specific details related thereof. The name of the one observation group is now obsgroup1 rather than tmf. The 87 observed data points are listed in the observation data section of the input control file, and they are all equally assigned a weight of one. The model command line section of the control file lists the model for this example and, as previously noted, it is *tp2.exe*. And the two relevant input and output files, also previously discussed, are listed in the model input/output section at the very end of the control file.

Step 06 – Verify control file is functional

One can check to see if there are any errors with the input control file as now prepared by typing the following at the command prompt and pressing enter:

PESTCHEK tp2_1

In so doing, one would see the following record of PESTCHEK execution at the command prompt (Figure 3):



```
C:\data\FY11_Work\CHL-GSSHA-MLSL\Examples\tp2>pestchek tp2_1

PESTCHEK Version 12.1.0. Watermark Numerical Computing.

Errors ----->
No errors encountered.

Warnings ----->
Consider setting JACUPDATE to 999 to implement Broyden's Jacobian updating.

C:\data\FY11_Work\CHL-GSSHA-MLSL\Examples\tp2>
```

Figure 3. Terminal display for PESTCHEK execution.

Step 07 – Modify control file for use with ERDC MLSL implementation

To employ our ERDC MLSL implementation, two additional rows of input data are appended to the end of the control data section of the control file *tp2_1.pst*, as shown below. The four entries on the first row appended (now, the next to last row of the control data section) to the end of the control data section of the control file are specific to our independent ERDC SLM implementation, and the description and function for each of the four entries is provided in Step 10 of Example 1 in our recent report (Skahill et al. 2011). So, as was indicated earlier in this report, with our MLSL implementation, one can either employ LM or SLM for local search. The first entry on the second row appended (now, the last row of the control data section) to the end of the control data section of the control file is a floating point value specifying a distance threshold that is used for comparison during execution of our implementation of MLSL. In particular, if during a given local search with MLSL, the distance computed between the current location in adjustable model parameter space and any of the previously computed parameter upgrade vectors, obtained either during the existing or with any of the previously performed local searches, is less than this specified threshold value, then the current local search is prematurely terminated with the assumption that it has progressed into a region of attraction of a previously visited local minimum. The impact of this input entry has not been explored with much detail, and it is often set to zero, as it is specified below. Further exploration for the specification of this input value is encouraged, not only for MLSL, but also for the stochastic GO method Trajectory Repulsion (Skahill and Doherty, 2006). The subsequent four entries on the second row appended (now, the last row of the control data section) to the end of the control data section of the control file are all specific to our ERDC MLSL implementation. Relative to our specification of our MLSL implementation outlined in Figure 1 above, the following four input values are *N*, γ, σ, and d_2, respectively. *N* is an integer input value; whereas, the remaining three are floating point values. *N*, γ, and σ impact the efficiency and effectiveness of MLSL. The second example in this report will explore their impact using the computationally efficient test problem from this first example.

```
* control data
restart estimation
 2 87 2 0 1
 1 1 single point 1 0 0
 5.0 2.0 0.3 0.03 10
100000.0 3.0 0.001 0
 0.1 noaui
30 0.001 4 4 0.001 4
 1 1 1
0 0 1.0 0
0 40 0.10 4 0.1
```

Step 08 – Calibrate test problem using ERDC MLSL implementation

Our implementation of MLSL was employed to calibrate the computationally efficient two parameter test problem, in a model independent manner, using the prepared input control file *tp2_1.pst* (and the related input files), by typing the following at the command prompt and pressing enter:

```
chl_mlsl
```

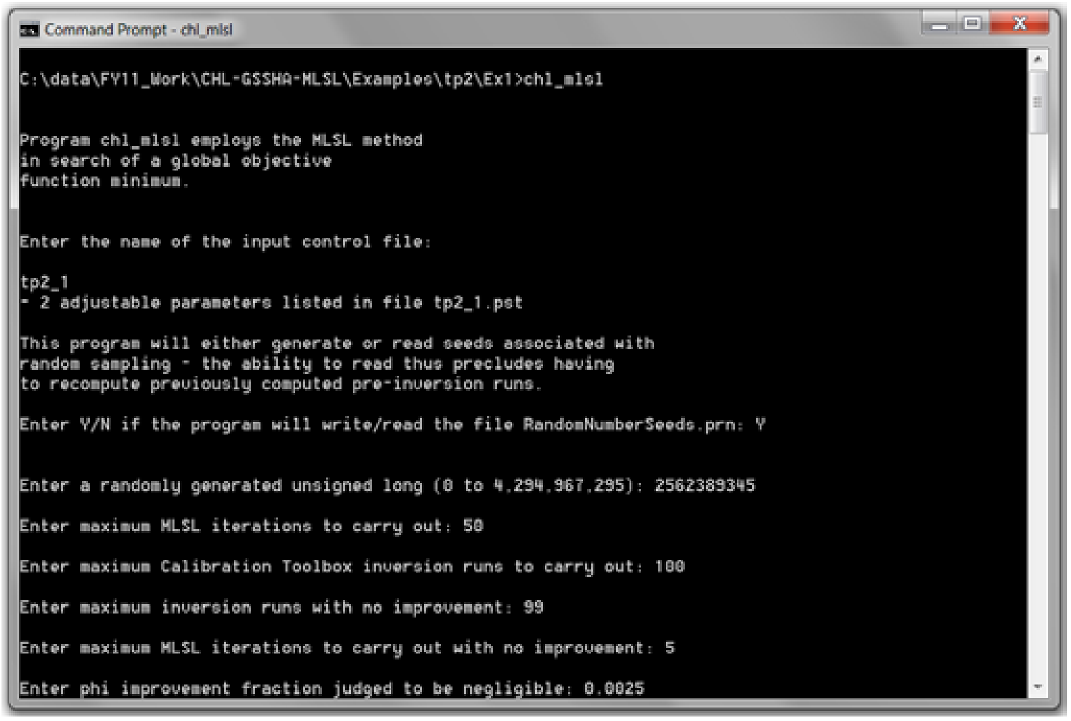
Upon execution of our current implementation of MLSL, the user is subsequently prompted for additional input parameters. Figure 4 summarizes its complete execution. In particular, upon execution of our implementation of MLSL, the user is asked for, in sequence,

1. The name of the modified input control file,
2. Whether the program is to write or read (thus enabling the reproduction of past efforts) the file named "*RandomNumberSeeds.prn*", which contains random numbers used during program execution,
3. An unsigned long that provides the basis for computation of the previously mentioned random numbers,
4. The maximum number of iterations to perform during MLSL execution,
5. The maximum number of local searches to perform during MLSL execution,
6. The maximum number of local searches to perform with no objective function improvement,
7. The maximum number of MLSL iterations to perform with no objective function improvement, and
8. The value which determines whether the previous and current computed best objective function value estimates are to be treated as effectively the same, or distinct.

The MLSL generated output file that will be of most interest to the user is named "*ct_mlsl.rec*", and its contents for this example are listed in Appendix 3. Examining its contents, we see that the file includes an echo of the MLSL input parameters, and then for each MLSL iteration,

1. A listing of the sampling phase,
2. The reduced sample set,
3. The value for the critical distance threshold (named alpha in the file),
4. A summary of the treatment of each member of the reduced sample set,

and then the file ends with a summary of the stochastic GO run, including the number of local searches that were performed, the number of local searches that yielded distinct local minima, the number of MLSL iterations, the minimum objective function value identified, at what point during MLSL execution the minimum objective function value was identified, the total number of forward model calls, and the reason for terminating MLSL execution. For example, with this computationally efficient two parameter test problem, MLSL, with the input as specified above and also listed in Appendix 3, identified the global minimum in the first iteration of its execution with the first local search, and it terminated execution after five additional MLSL iterations (and one additional local search that yielded another local minimum) yielded no further objective function improvement. One hundred and seventy-six forward model calls were required to find the global minimum.



```
C:\data\FY11_Work\CHL-GSSHA-MLSL\Examples\tp2\Ex1>chl_mlsl


Program chl_mlsl employs the MLSL method
in search of a global objective
function minimum.


Enter the name of the input control file:

tp2_1
- 2 adjustable parameters listed in file tp2_1.pst

This program will either generate or read seeds associated with
random sampling - the ability to read thus precludes having
to recompute previously computed pre-inversion runs.

Enter Y/N if the program will write/read the file RandomNumberSeeds.prn: Y


Enter a randomly generated unsigned long (0 to 4,294,967,295): 2562389345

Enter maximum MLSL iterations to carry out: 50

Enter maximum Calibration Toolbox inversion runs to carry out: 100

Enter maximum inversion runs with no improvement: 99

Enter maximum MLSL iterations to carry out with no improvement: 5

Enter phi improvement fraction judged to be negligible: 0.0025
```

Figure 4. Terminal display for MLSL method (example 1).

Example 2

In this example, we will further examine MLSL execution using the computationally efficient two parameter test problem that was presented in the previous example. In particular, we will examine the impact of the input parameters N, γ, and σ on the performance of MLSL; viz., its efficiency (summarized by the average and standard deviation of total model calls for 100 MLSL trial runs) and effectiveness (summarized by the number of failures to find the global minimum among 100 MLSL trial runs). We will do so by performing 100 trials for 13 unique *N*, γ, and σ permutations. Since all 13 100 MLSL trial runs are very similar in terms of execution, we will discuss how to do, by way of example, one instance, and then summarize the results for all 13 100 MLSL trial runs in the table we present below.

Step 01 – Prepare 100 MLSL input files to automate execution of 100 MLSL trial runs

To enable the automatic execution of 100 MLSL trial runs, 100 MLSL input files were prepared whose contents are like what is shown below. The only difference between each separate input file is the third input value within each file; viz., the unsigned long that provides the basis for computation of the random numbers which impact the sampling phase which occurs in each iteration of MLSL execution. One hundred files were prepared with contents like what is shown below and the 100 files were numbered in the following manner: tp2_1_1.in, tp2_1_2.in, tp2_1_3.in, …, tp2_1_99.in, and tp2_1_100.in.

```
tp2_1
Y
4063522982
50
100
99
5
0.0025
```

Step 02 – Prepare batch file for execution of multiple MLSL runs

A batch file named chl_mlsl_all.bat was prepared and its contents are listed directly below. It supports the execution of multiple MLSL runs.

```
@ECHO ON
IF [%1]==[] GOTO INPUTERROR
::
Set Count=
::
:LOOP {The count advances one each time through the LOOP}
::
::
```

```
:: Start Counted process.
::
Set /A Count += 1
@echo %Count%
::
::
chl_mlsl tp2_1.pst < tp2_1_%Count%.in
copy RandomNumberSeeds.prn RandomNumberSeeds_%Count%.prn
copy ct_mlsl.rec ct_mlsl_%Count%.rec
copy tp2_1.rec tp2_1_%Count%.rec
::
::
IF NOT [%Count%]==[%1] GOTO LOOP
::
Goto END
::
:INPUTERROR
@ECHO USAGE: TPL {Number of runs}
@ECHO.
::
:END
```

Step 03 – Calibrate model 100 times with 100 unique calls to MLSL

The computationally efficient two parameter test problem was calibrated 100 times by way of 100 MLSL trial runs that were performed by typing the following at the command prompt and pressing enter:

```
chl_mlsl_all 100
```

Table 1 summarizes the results from all 13 separate (calibration) cases of 100 MLSL trial global optimization runs. The observed differences for each case are completely a function of the MLSL input parameters *N*, γ, and σ.

Table 1. Results from all 13 separate cases of 100 MLSL trial global optimization runs.

Case	1	2	3	4	5	6	7	8	9	10	11	12	13
MLSL Input Parameters													
N	40	40	40	40	20	20	10	10	10	10	20	20	20
γ	0.05	0.1	0.05	0.05	0.05	0.05	0.1	0.2	0.1	0.1	0.05	0.05	0.05
σ	4	4	2	1	4	2	4	4	2	1	8	1	16
Summary of 100 MLSL Trial Runs													
Average # of total model calls	414.66	612.24	421.25	436.59	267.7	275.93	245.29	347.43	288.3	306.11	229.87	286.52	206.47
Standard deviation for total model calls	80.05	142.88	84.98	92.64	82.32	78.98	100.62	124.89	101.65	103.65	75.82	84.53	66.89
Number of failures to find global minimum	0	0	0	0	0	0	4	3	1	1	1	0	2

Example 3

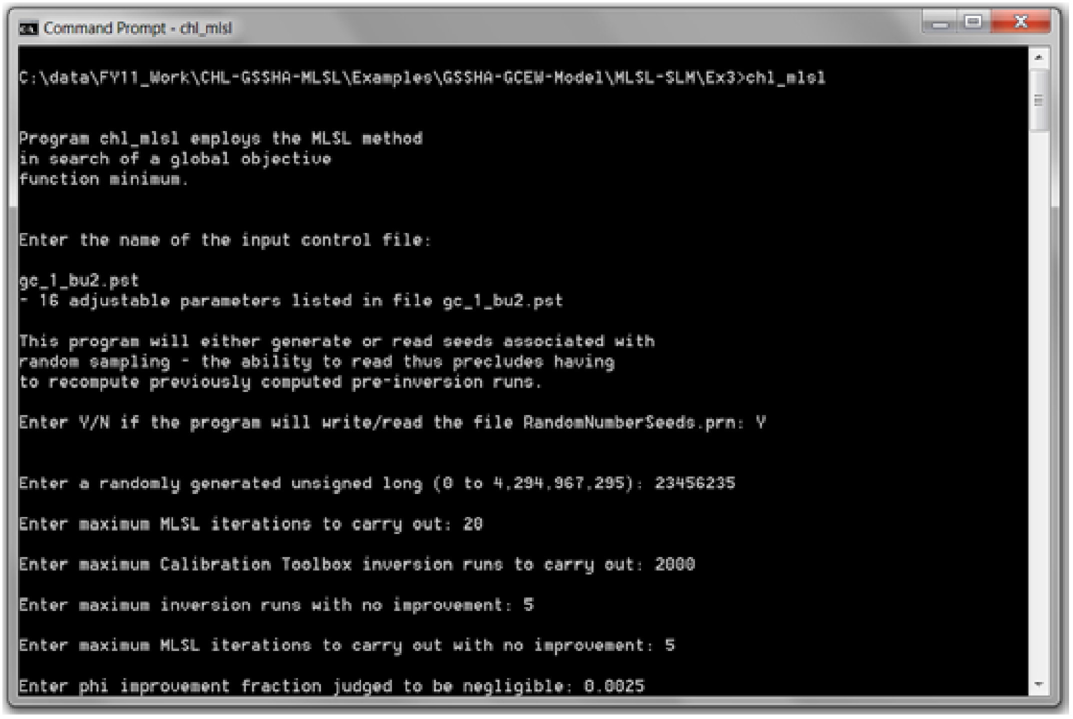
In this example, we will describe the steps necessary to perform a MLSL GO run to calibrate a GSSHA hydrologic model for the GCEW. The noted GSSHA GCEW hydrologic model is the same model that was also considered in the examples in our recent report which documented how to use our independent implementations of the LM/SLM local search methods for model calibration (Skahill et al. 2011). To that end, before continuing any further with this example, wherein we will discuss how to use the stochastic GO method MLSL to calibrate the GSSHA GCEW hydrologic model, the active reader is first directed to complete the first and third examples in our recent report (Skahill et al. 2011). In particular, the point of departure, relative to completion of example 1 and example 3 from our recent report, and our intent to use MLSL, is to simply augment the control data section of the input control file from example 3 in our recent report (Skahill et al. 2011); viz., the input control file *gc_1_bu2.pst*, as shown below. The four new additions, which are relevant to MLSL execution, are highlighted yellow, and the one edit, which most likely will effectively not allow there to be a full update of the model Jacobian during any of the local searches, is highlighted orange.

```
* control data
restart estimation
 16 233 16 0 1
 1 1 single point 1 0 0
5.0 2.0 0.3 0.03 10
5.0 5.0 1.0e-3
0.1 noaui
30 .005 4 4 .005 4
1 1 1
1 0 100000000000.0 0
0 20 0.05 4 0.1
```

After the slight alteration to the input control file noted above, all that is now required is to call MLSL at the command prompt

```
chl_mlsl
```

The inputs that were further specified at the command prompt for MLSL execution are shown below (Figure 5). And the contents of the file *ct_mlsl.rec*, which summarize the salient aspects of the MLSL GO run to calibrate the GSSHA GCEW hydrologic model, are listed in Appendix 4.

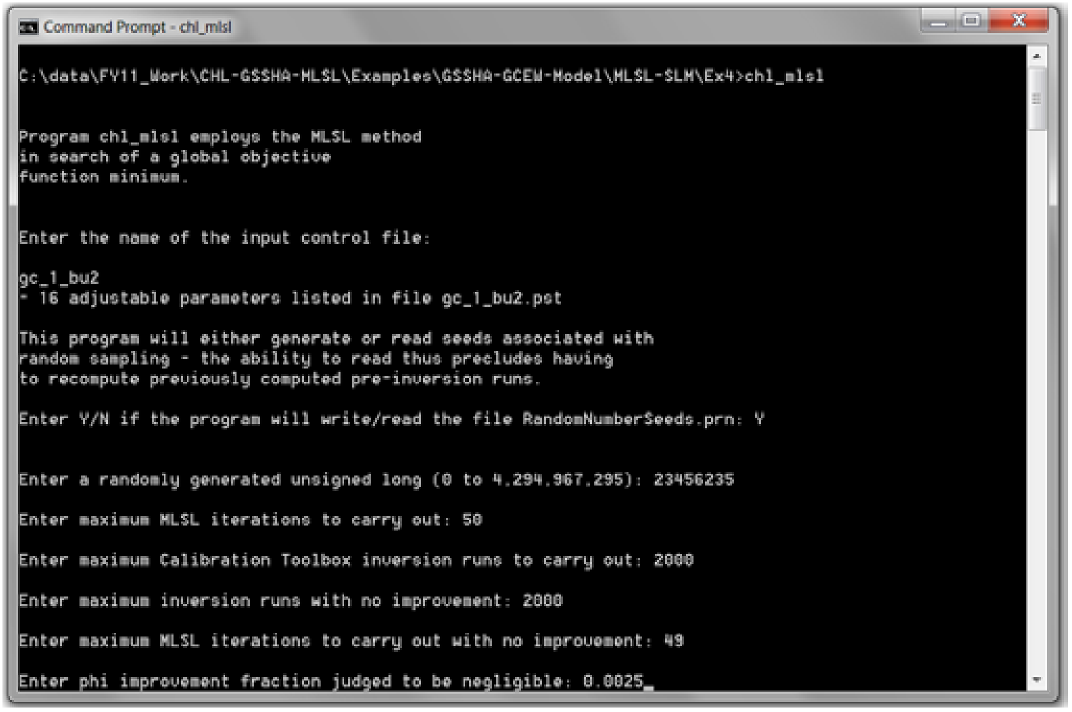


```
C:\data\FY11_Work\CHL-GSSHA-MLSL\Examples\GSSHA-GCEW-Model\MLSL-SLM\Ex3>chl_mlsl


Program chl_mlsl employs the MLSL method
in search of a global objective
function minimum.


Enter the name of the input control file:

gc_1_bu2.pst
- 16 adjustable parameters listed in file gc_1_bu2.pst

This program will either generate or read seeds associated with
random sampling - the ability to read thus precludes having
to recompute previously computed pre-inversion runs.

Enter Y/N if the program will write/read the file RandomNumberSeeds.prn: Y


Enter a randomly generated unsigned long (0 to 4,294,967,295): 23456235

Enter maximum MLSL iterations to carry out: 20

Enter maximum Calibration Toolbox inversion runs to carry out: 2000

Enter maximum inversion runs with no improvement: 5

Enter maximum MLSL iterations to carry out with no improvement: 5

Enter phi improvement fraction judged to be negligible: 0.0025
```

Figure 5. Terminal display for MLSL method (example 3).

Example 4

A second calibration run for the GSSHA hydrologic model for the GCEW was also performed using our MLSL implementation, this time with the MLSL input parameter σ set equal to one rather than four, as in the previous (third) example, in attempts to require more exploration of adjustable model parameter space. The termination criteria for this example, relative to the previous (third) example, were specified at the command prompt in a manner to encourage a more exhaustive search than what was performed in the third example (Figure 6). The contents of the file *ct_mlsl.rec*, which summarize the salient aspects of the MLSL GO run to calibrate the GSSHA GCEW hydrologic model, are listed in Appendix 5.



```
C:\data\FY11_Work\CHL-GSSHA-MLSL\Examples\GSSHA-GCEW-Model\MLSL-SLM\Ex4>chl_mlsl


Program chl_mlsl employs the MLSL method
in search of a global objective
function minimum.


Enter the name of the input control file:

gc_1_bu2
- 16 adjustable parameters listed in file gc_1_bu2.pst

This program will either generate or read seeds associated with
random sampling - the ability to read thus precludes having
to recompute previously computed pre-inversion runs.

Enter Y/N if the program will write/read the file RandomNumberSeeds.prn: Y


Enter a randomly generated unsigned long (0 to 4,294,967,295): 23456235

Enter maximum MLSL iterations to carry out: 50

Enter maximum Calibration Toolbox inversion runs to carry out: 2000

Enter maximum inversion runs with no improvement: 2000

Enter maximum MLSL iterations to carry out with no improvement: 49

Enter phi improvement fraction judged to be negligible: 0.0025_
```

Figure 6. Terminal display for MLSL method (example 4).

3 Results and Discussion

Example 2 explored the impact of the MLSL input parameters N, γ, and σ on overall algorithm efficiency and effectiveness by considering thirteen separate cases of 100 MLSL global optimization trial runs which were directed at calibrating the computationally efficient two parameter test problem that was originally presented in example 1. Examining the table that was presented in example 2 (Table 1), which summarized the thirteen separate numerical experiments involving 100 trials in each case, the following observations are given:

1. With γ and σ fixed, as N increases, so does the average for the total number of forward model calls.
2. With N and γ fixed, as σ decreases, we observe that the average and standard deviation for the total number of forward model calls both increase, modestly, and also that the effectiveness of the MLSL algorithm improves. For example, we see that with N, γ, and σ equal to 20, 0.05, and 16, respectively, we have 2 failures for the 100 trials, but only 1 failure for the 100 trials when $\sigma = 8$, and no failures for the 100 trials when $\sigma = 4$, 2, or 1. The MLSL algorithm is more effective with the lower σ values because the critical distance which is used to characterize the size of the clusters is a function of σ (see Figure 1). At a lower σ value, cluster size is smaller; hence, the opportunity for a single cluster to contain multiple regions of attraction is reduced.
3. With N and σ fixed, as γ increases so does the average and standard deviation for the total number of forward model calls, simply because γ controls the potential number of local searches to be performed during each iteration of MLSL execution, and as γ increases so does that potential.

Based on our own experience to date, we recommend that users of our implementation of MLSL start with a value of one for σ, and choose values for N and γ that take into consideration the expense associated with a single forward model call and the understanding that MLSL is an iterative stochastic GO method. And we recommend that termination criteria be specified in a manner consistent with the level of importance assigned to finding the global minimizer for the given application.

In examples 3 and 4 we used our implementation of MLSL to calibrate the GSSHA hydrologic model for the GCEW which was originally calibrated as part of the work effort documented in our recent report using our independent LM/SLM local search method implementations. For examples 3 and 4, in consideration of the expense associated with a single forward model call for the GSSHA GCEW hydrologic model, we used the SLM method for local search, in such a manner that no measures were employed to mitigate against the potential that our approximation to the true model Jacobian may become poor after some optimization iterations. In example 3, 517 total model calls were required to calibrate the model using MLSL with the given input parameters (as specified above and also at the very beginning of Appendix 4). And in example 4, with the value for σ reduced relative to its value used for example 3 (from 4 in example 3 to one in example 4), and also using stricter termination criteria, 3832 total model calls were required to calibrate the GSSHA GCEW model using MLSL with the given input parameters (as specified above and also at the very beginning of Appendix 5). As a result, in example 3, 7 local searches were performed and 7 distinct local minima were identified; whereas, in example 5, 55 local searches were performed and 55 distinct local minima were identified. With example 3, the minimum and maximum identified objective function values were 58.126056 and 70.299196, respectively, and the minimum value was found from the fourth of the seven total local searches. With example 4, the minimum and maximum identified objecttive function values were 57.825337 and 77.795517, respectively, and the minimum value was found from the fiftieth of the fifty-five total local searches. Figures 7 and 8 are plots of the transformed observed and simulated flows that constitute the minimum objective function values that were identified in Examples 3 and 4, respectively. The figures provide the interested reader with a means to effectively compare the minimum objective function values that were obtained in the two examples in terms of a statistical summary (i.e., R^2 and Nash-Sutcliffe efficiency score) and also by way of a scatter plot of the transformed observed data and its model simulated counterparts. Clearly, there is little difference in terms of fit with the observed data between the models associated with the minimum objective function values that were identified in Example 3 and 4. The salient difference is that the minimum identified in Example 4 required approximately an order of magnitude more model calls. Possibly even of greater interest to the engaged reader would be to compare the results presented in Figures 7 and 8 with the results that were displayed in a similar manner for Example 3 in our recent report (Skahill et al. 2011), wherein the same GSSHA GCEW hydrologic model was calibrated using the independent ERDC SLM implementation, employed in the same manner as it was for the local searches within MLSL in Examples 3 and 4. Performing such a comparison, one would see comparable results were obtained when using SLM simply once, with no cyclic updating, in terms of the fit achieved between the observed data and their model simulated counterparts, wherein an associated final objective function value of 59.56 was obtained in only 62 total model calls.

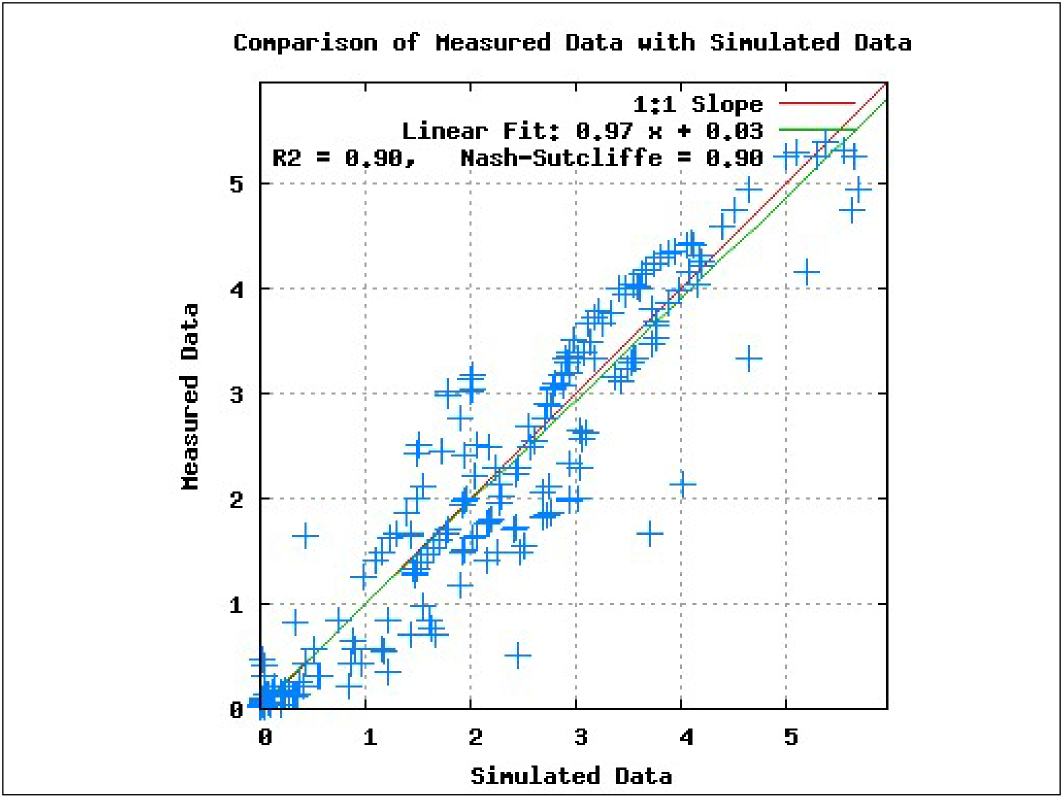


Figure 7. Plot of transformed observed and simulated flows, associated with the minimum objective function value identified in Example 3.

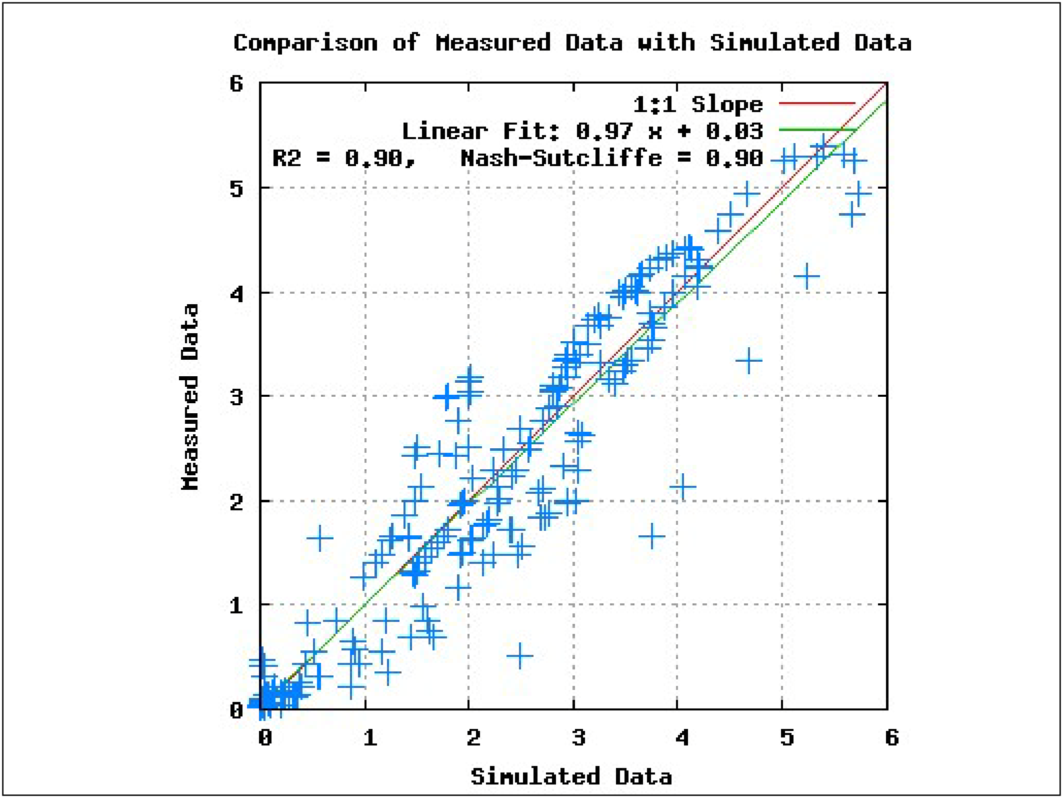


Figure 8. Plot of transformed observed and simulated flows, associated with the minimum objective function value identified in Example 4.

4 Summary

In this report, we demonstrated how to use our implementation of the stochastic global optimization technique called Multi Level Single Linkage (MLSL), which uses our LM/SLM method implementations for local search. We considered four separate examples to demonstrate its practical use. All four examples emphasized how simple it is to use our MLSL implementation after a given forward model has already been interfaced with our LM/SLM software for local search (see our recent report (Skahill et al. 2011)). The first example discussed all of the steps necessary to use our MLSL implementation to calibrate a computationally efficient two parameter test problem. The second example also involved calibrating the computationally efficient two parameter test problem that was considered in example 1. It considered a series of 13 separate numerical experiments, each involving 100 unique MLSL runs for a given set of input parameters, in attempts to provide the active reader with an intuitive feel for how the principal MLSL input parameters affect MLSL algorithm efficiency and effectiveness. The third and fourth examples both involved demonstrating how to calibrate, using our MLSL implementation, the same GSSHA GCEW hydrologic model that we also considered in our recent report (Skahill et al. 2011) wherein we demonstrated how to use our LM/SLM local search implementations for GSSHA hydrologic model calibration. The fourth experiment was designed, among others, to provide an opportunity to examine MLSL termination criteria and their impact on overall algorithm execution. We provided some initial guidance regarding the specification of MLSL input parameter values and termination criteria.

Our quick comparison between the results that were obtained in Examples 3 and 4, and also with the SLM local search calibration run that was performed and documented upon in our recent report (Skahill et al. 2011) encouraged us, based on our limited applications to date, to recommend that GSSHA hydrologic modelers currently use our SLM implementation, possibly with prior information, rather than MLSL for GSSHA hydrologic model calibration. However, if needed and/or desired, then we recommend MLSL as the global optimization tool to use to calibrate GSSHA, rather than the SCE method, which is currently employed for automatic calibration of GSSHA models (please see Skahill et al. (2009) for further details). The previously mentioned comparisons underscore the need to provide for an effective, efficient, and adaptive means for calibrating spatial hydrology models such as GSSHA. We see that to be possible by merging ideas from previous contributions, in particular, by merging ideas from Skahill et al. (2009) and Doherty and Skahill (2006). This is our current research and development focus for GSSHA hydrologic model calibration.

The user of our ERDC implementation of the MLSL method accepts and uses the software at his/her own risk. Any questions, comments, and/or concerns regarding the use of our ERDC software implementation of the MLSL method with the GSSHA model should be directed to the first author. The interested reader is encouraged to contact the second author with any questions, comments, and/or concerns related to the GSSHA hydrologic simulation model.

References

Baginska, B., W. Milne-Home, P. S. Cornish. 2003. Modelling nutrient transport in Currency Creek, NSW with AnnAGNPS and PEST. Environmental Modelling & Software 18, 801-808.

Bicknell, B. R., J. C. Imhoff, J. L. Kittle, T. H. Jobes, A. S. Donigian. 2001. HSPF User's Manual. Aqua Terra Consultants, Mountain View, California.

DeGroote-Hedlin, C., and S. Constable. 1990. Occam's inversion to generate smooth, two-dimensional models from magnetotelluric data. Geophysics, 55(12), 1631-1624.

Doherty, J. 2004. PEST: Model Independent Parameter Estimation. Fifth edition of user manual. Watermark Numerical Computing, Brisbane, Australia.

Doherty, J. 2007a. Addendum to the PEST Manual. Watermark Numerical Computing, Brisbane, Australia, January.

Doherty, J. 2007b. Addendum to the PEST Manual. Watermark Numerical Computing, Brisbane, Australia, August.

Doherty, J. 2007c. PEST Surface Water Utilities. Watermark Numerical Computing, Brisbane, Australia.

Doherty, J., and J. M. Johnston. 2003. Methodologies for calibration and predictive analysis of a watershed model. J. American Water Resources Association, 39(2), 251-265.

Doherty, J., and B. E. Skahill. 2006. An Advanced Regularization Methodology for Use in Watershed Model Calibration. Journal of Hydrology, 327, 564– 577.

Downer, C. W., and F. L. Ogden. 2003a. GSSHA user's manual: Gridded surface subsurface hydrologic analysis, version 1.43 for WMS 6.1, Technical Report, U.S. Army Corps of Eng., Eng. Res. and Dev. Cent., Vicksburg, Miss..

Downer, C. W., and F. L. Ogden. 2003b. Prediction of runoff and soil moistures at the watershed scale: Effects of model complexity and parameter assignment. Water Resources Research, Vol. 39, No. 3, 1045.

Duan, Q. S., S. Sorooshian, and V. K. Gupta. 1992. Effective and efficient global optimization for conceptual rainfall runoff models. Water Resour. Res. 28 (4), 1015-1031.

Duan, Q., V. K. Gupta, and S. Sorooshian. 1993. A Shuffled Complex Evolution approach for effective and efficient global minimization. Journal of Optimization Theory and its Applications, 76 (3), 501-521.

Frankenstein, S., and G. Koenig. 2004. Fast All-season Soil STrength (FASST). U.S. Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory, Hanover, New Hampshire, Special Report SR-04-1.

Gallagher, M., and J. Doherty. 2007a. Predictive error analysis for a water resource management model. Journal of Hydrology, (334), 513–533.

Gallagher, M., and J. Doherty. 2007b. Parameter estimation and uncertainty analysis for a watershed model. Environmental Modelling & Software 22, 1000-1020.

Goegebeur, M., and V. R. N. Pauwels. 2007. Improvement of the PEST parameter estimation algorithm through Extended Kalman Filtering. Journal of Hydrology, (337), 436-451.

Gutiérrez-Magness, A. L., and R. H. McCuen. 2005. Effect of Flow Proportions on HSPF Model Calibration Accuracy. J. Hydrologic Engrg. 10, 343-352.

Hansen, N., and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. Evol. Comput., 9, 159-195.

Hansen, N., S. D. Muller, and P. Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput, 9, 159-195.

Haydon, S., and A. Deletic. 2007. Sensitivity testing of a coupled Escherichia coli-Hydrologic catchment model. Journal of Hydrology, (338), 161-173.

Iskra, I., and R. Droste. 2007. Application of Non-Linear Automatic Optimization Techniques for Calibration of HSPF. Water Environment Research, 79 (6), 647-659.

Kim, S. M., B. L. Benham, K. M. Brannan, R. W. Zeckoski, and J. Doherty. 2007. Comparison of hydrologic calibration of HSPF using automatic and manual methods. Water Resour. Res., 43, W01402, doi:10.1029/2006WR004883.

Kunstmann, H., J. Krause, and S. Mayr. 2006. Inverse distributed hydrological modelling of Alpine catchments. Hydrology and Earth System Sciences. 10 (3), 395-412.

Levenberg, K. 1944. A method for the solution of certain problems in least squares. Q. Appl. Math., 2, 164-168.

Maneta, M. P., G. B. Pasternack, W. W. Wallender, V. Jetten, and S. Schnabel. 2007. Temporal instability of parameters in an event-based distributed hydrologic model applied to a small semiarid catchment. Journal of Hydrology, Volume 341, 207-221.

Marquardt, D. 1963. An algorithm for least-squares estimation of non-linear parameters. SIAM J. Appl. Math., 11, 431-441.

Menke, W. 1984. Geophysical Data Analysis. Academic Press Inc. Orlando, Florida.

Moore, C., and J. Doherty. 2005. The role of the calibration process in reducing model predictive error. Water Resources Research. Vol 41, No 5. W05050.

Rinnooy Kan, A. H. G, and G. Timmer. 1987a. Stochastic Global Optimization Methods, Part I: Clustering Methods, Mathematical Programming, 39, 27-56.

Rinnooy Kan, A. H. G, and G. Timmer. 1987b. Stochastic Global Optimization Methods, Part II: Multi Level Methods, Mathematical Programming, 39, 57-78.

Rode, M., U. Suhr, and G. Wriedt. 2007. Multi-objective calibration of a river water quality model--Information content of calibration data. Ecological Modelling, 204, 129-142.

Rose, K. A., B. A. Megrey, F. E. Werner, and D. M. Ware. 2007. Calibration of the NEMURO nutrient-phytoplankton-zooplankton food web model to a coastal ecosystem: Evaluation of an automated calibration approach. Ecological Modelling, (202), 38-51.

Santanello Jr., J. A., C. D. Peters-Lidard, M. E. Garcia, D. M. Mocko, M. A. Tischler, M. S. Moran, and D. P. Thoma. 2007. Using remotely-sensed estimates of soil moisture to infer soil texture and hydraulic properties across a semi-arid watershed. Remote Sensing of Environment, (110), 79-97.

Senarath, S. U. S., F. L. Odgen, C. W. Downer, and H. O. Sharif. 2000. On the calibration and verification of two-dimensional, distributed, Hortonian, continuous watershed models. Water Resources Res., 36 (6), 1495-1510.

Skahill, B., and J. Doherty. 2006. Efficient accommodation of local minima in watershed model calibration. Journal of Hydrology, (329), 122-139.

Skahill, B., J. Baggett, S. Frankenstein, and C. W. Downer. 2009. Efficient Levenberg-Marquardt Method Based Model Independent Calibration. Environmental Modelling & Software (24), 517–529.

Skahill, B., C. W. Downer, and J. Baggett. 2011. A Practical Guide to Calibration of a GSSHA Hydrologic Model using ERDC Automated Calibration Software – Efficient Local Search. ERDC Technical Report 11-XX. (in press)

Tonkin, M., and J. Doherty. 2005. A hybrid regularised inversion methodology for highly parameterised models. Water Resources Research. Vol. 41, W10412, doi:10.1029/2005WR003995.

Törn, A., and A. Žilinskas. 1987. Global Optimization. Lecture Notes in Computer Science, 350, Springer-Verlag.

Zyvoloskia, G., E. Kwicklis, A. A. Eddebbarh, B. Arnold, C. Faunt, and B. A. Robinson. 2003. The site-scale saturated zone flow model for Yucca Mountain: calibration of different conceptual models and their impact on flow paths. Journal of Contaminant Hydrology, 62–63, 731–750.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE ***(DD-MM-YYYY)*** February 2012	**2. REPORT TYPE** Final Report	**3. DATES COVERED** ***(From - To)***
4. TITLE AND SUBTITLE A practical guide to calibration of a GSSHA hydrologic model using ERDC automated model calibration software – effective and efficient stochastic global optimization		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Brian E. Skahill, Charles W. Downer, and Jeffrey S. Bagget		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Engineer Research and Development Center Coastal and Hydraulics Laboratory 3909 Halls Ferry Road; Vicksburg, MS 39180; University of Wisconsin at La C rosse 1725 State Street; La Cross,WI 54601		**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CHL TR-12-2
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Corps of Engineers 441 G. Street, NW Washington, DC 20314-1000		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The objective of this article is to demonstrate, by way of example(s), how to use our implementation of the MLSL method for model independent parameter estimation to calibrate a GSSHA hydrologic model. The purpose is not to present or focus on the theory which underlies the parameter estimation method, but rather to carefully describe how to use the ERDC software implementation of MLSL that accommodates the PEST model independent interface to calibrate a GSSHA hydrologic model. Given the computational expense associated with global optimization, we will initially consider variations of our MLSL implementation on a computationally efficient test problem in attempts to provide the interested reader with an intuitive sense of how the method works.

15. SUBJECT TERMS
Calibration
Global optimization
GSSHA
Multi-Level Single Linkage

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED		35	**19b. TELEPHONE NUMBER** ***(include area code)***