# A Statistical Approach for Calibrating Hydrologic Models

by

Natalia Bhattacharjee (nèe Shim)

(Under the Direction of Abhyuday Mandal)

## Abstract

This work is an interdisciplinary research involving the fields of statistics and hydrology. In hydrology, rainfall-runoff models give a wide spectrum of output series and calibration procedures require significant amount of time. Calibration of these hydrologic models especially is a challenging task since the input parameters of these models are often unknown and correspond to physical properties that are difficult to measure. In statistics, model parameter estimation (calibration) problem simplifies to finding an inverse solution of a computer model that generates time series output. In this research, we focus on solving the inverse problem for hydrologic time series and, thus, calibrating the computer model. We propose a modified history matching approach for calibrating rainfall-runoff models efficiently. We present the methodology and illustrate the application of the algorithm using both synthetic and field data (one simulation study and two case studies). We calculate several goodness-of-fit statistics to assess the performance of the modified history matching algorithm. The results demonstrate that the proposed approach improved model performance by 30 % and 11 % in the case studies of compartment model and SWAT model, respectively.

Index words: Computer Experiments, Contour Estimation, Inverse Problem, Emulator, Simulator, Gaussian Process Model, Time Series, Hydrology

# A Statistical Approach for Calibrating Hydrologic Models

by

Natalia Bhattacharjee (nèe Shim)

Integrated BS and MS, Omsk State Technical University, 2006

PhD, University of Georgia, 2017

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2018

# A Statistical Approach for Calibrating Hydrologic Models

by

Natalia Bhattacharjee (nèe Shim)

# A Statistical Approach for Calibrating Hydrologic Models

Natalia Bhattacharjee (nèe Shim)

April 23, 2018

# Dedication

This thesis is dedicated to my family and my husband for providing their unconditional love, support and motivation.

# Acknowledgments

I would like to thank my advisor, Dr. Abhyuday Mandal, for providing his guidance, sharing his expertise and research insights throughout my journey of pursuing M.S. degree in Statistics at the University of Georgia (UGA). I would like to thank my committee members, Dr. Pritam Ranjan and Dr. Ernest W. Tollner, for their suggestions, insightful comments, patience and time. I also want to thank Dr. Cheolwoo Park, Dr. Nicole Lazar, Dr. Mark Werner, Dr. Jaxk Reeves, for their help and discussions.

I would like to thank Georgia Water Resource Institute (GWRI) for the opportunity to participate in the State Water Resources Research Institute (WRII) Program supported by U.S. Geological Survey (USGS) and for awarding Research Grant 2015 – 2017 and partially supporting this research. Also, I am thankful to the Center of Geographical Research at UGA for the opportunity to participate in NASA DEVELOP National Program and to gain more knowledge and experience in working with geospatial analysis, earth science data and remote sensing techniques. Additionally, I thank Dr. Nathan and staff in Bioconversion Center for sharing long-term time series data of Windrow Composting Pad operation.

I am so grateful to my parents who have been sharing this journey from a long distance and supporting me in studying and working abroad for more than ten years. I am very thankful for all their patience and love. And last, but not least, to my husband and life partner, Tapomayukh Bhattacharjee, for his encouragement and for believing in me. I cannot thank him enough for his continuous support, love and care; and for being there when I needed most.

# Contents

**1 Rainfall-Runoff Models in Hydrology 1**

1.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Case study 1: Matlab-Simulink model . . . . . . . . . . . . . . . . . . . 2

1.3 Case study 2: SWAT model . . . . . . . . . . . . . . . . . . . . . . . . 6

1.4 Calibration Problem . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

1.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

**2 Calibration Problem in Statistics 12**

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.2 Gaussian Process Model . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.3 Latin Hypercube Design . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.4 Expected Improvement Criterion . . . . . . . . . . . . . . . . . . . . . . 18

2.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

**3 Review of Existing Algorithms 20**

3.1 Compartment Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

3.2 SWAT and SUFI-2 Algorithm . . . . . . . . . . . . . . . . . . . . . . . 22

3.3 Scalarization Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . 25

**4 Proposed History Matching Approach 29**

4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

4.2 History Matching Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . 30

4.3 Modified History Matching Algorithm . . . . . . . . . . . . . . . . . . . . 32

4.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

**5 Performance of the Proposed Algorithm 40**

5.1 Case study 1: Matlab-Simulink Model Calibration . . . . . . . . . . . . . . 40

5.2 Case study 2: SWAT Model Calibration . . . . . . . . . . . . . . . . . . . 42

5.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

**References 44**

**Appendices 50**

**A MATLAB code for the case study 51**

**B R code for the case study 60**

**C R code for the simulation study 67**

# List of Figures

1.1 Google Earth (© 2010) overhead view of the windrow composting pad at the Bioconversion Center, UGA, Athens (Duncan *et al.*, 2013b). . . . . . . . . . . . . 3

1.2 Discharge field data ($g_0(t)$) from Bioconversion center, UGA (represented by the red curve) and the Matlab-Simulink model discharge outputs $g(x, t)$ (represented by the blue lines) for $t = 1, 2, ..., 5445$ at randomly generated $x$ (depth of surface, depth of sub-surface and two coefficients of the saturated hydraulic conductivity $K_{sat1}$ and $K_{sat2}$ ). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 Schematic of cross-sectional view of composting pad at the Bioconversion center, UGA, Athens, USA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.4 Map of the Middle Oconee River near Ben Burton Park, *Google Maps*. . . . . . . . 7

1.5 SWAT input data sources . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

1.6 Middle Oconee river discharge field data (USGS gauge number 02217500), $g_0(t)$ (represented by the red curve), and SWAT model discharge outputs $g(x, t)$ (represented by the blue curves) for $t = 1, 2, ..., 84$ at randomly generated $x$ (Manning's coefficient for the main channel, effective hydraulic conductivity in the main channel, groundwater delay, groundwater "revap" coefficient and available water capacity). 8

1.7 Conceptual representation of SWAT model. . . . . . . . . . . . . . . . . . . . 9

1.8 Sensitivity Analysis procedure in SWAT CUP. . . . . . . . . . . . . . . . . . . 10

2.1 Illustration of computer experiments . . . . . . . . . . . . . . . . . . . . . . 13

2.2 Illustration of Latin Hypercube Design . . . . . . . . . . . . . . . . . . . . . . . 17

3.1 Flowchart of key steps in scalarization method . . . . . . . . . . . . . . . . . . . 26

3.2 Calibration of the illustrative example using the scalarization method with $n_0 = 10$, $N = 40$, $M = 5000$ and the GP model as the surrogate. . . . . . . . . . . . . . . . 28

4.1 The illustrative example: a few model outputs (dashed curves) and the target response (solid curve). . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

4.2 The illustrative example: selection of the training points according to the implausibility function at the discretization-point-set $DPS = (33, 67)$ in the modified HM algorithm. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

4.3 The illustrative example: marginal distribution of the median of the minimized $\log[d'(x_i)]$ over 100 simulations for different two-factor combinations of $n_1, c, T_k$ and $M$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

4.4 The illustrative example: Sensitivity of selecting DPS measured with respect to the total run-size and optimized $\log[d'(x_i)]$. . . . . . . . . . . . . . . . . . . . . . 38

5.1 Calibration Results for the Matlab-Simulink model. The solid red curve represents observed data, blue dash line represents best solution used in the previous study and green dash line corresponds to the best solution using the proposed history matching algorithm. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

5.2 SWAT model calibration: The solid red curve represents the observed data, blue dashed line represents best solution using SUFI2 and green dashed line corresponds to the best solution using the HM algorithm. . . . . . . . . . . . . . . . . . . . . 42

# List of Tables

1.1 Summary of data processing for SWAT . . . . . . . . . . . . . . . . . . . . . . . 7

3.1 Full set of SWAT parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

5.1 NSE, $R^2$ and RMSE comparisons for Matlab-Simulink Model . . . . . . . . . . . 41

5.2 NSE, $R^2$ and RMSE comparisons for SWAT model . . . . . . . . . . . . . . . . . 43

# Chapter 1

# Rainfall-Runoff Models in Hydrology

In this chapter, we focus on description of the rainfall-runoff models. Section 1.1 provides a brief introduction to hydrologic models. Section 1.2 provides details on the compartment model for estimating rainfall-runoff relationship for the windrow composting pad. Section 1.3 describes Soil and Water Assessment Tool (SWAT) model and its data sources. Section 1.4 explains the importance of solving the calibration problem for hydrologic models.

## 1.1 Introduction

Hydrologic models are commonly used in environmental studies to estimate the water cycle elements in an area of interest and explain/predict complex physical processes. These models apply basic principles of mass balance, energy conservation and other principles of physics. In hydrology, numerous studies have been focused on the topic of modeling the rainfall-runoff relationship. In this work, we present two case studies referred to rainfall-runoff models (computer simulators).

The first case study represents Matlab-Simulink compartmental dynamic model that estimates the amount of runoff from a windrow composting pad (Duncan *et al.*, 2013b). The second case study includes the Soil and Water Assessment Tool (SWAT) model, a complex hydrologic model that simulates runoff from watershed areas based on climate variables, soil types, elevation and land use data (Arnold *et al.*, 1994).

## 1.2 Case study 1: Matlab-Simulink model

### Introduction

The first case study deals with Matlab-Simulink model which simulates runoff from windrow compost pad over a period of time. Composting is one of the effective methods of organic material decomposition (Kalaba *et al.*, 2007). The composting pad runoff must be collected into a pond prior to its further use since the composting systems are usually located in remote areas (Dorahy *et al.*, 2009). Curve number method is traditionally used for predicting runoff from rainfall events. But this method is very site specific and not realistic for forecasting runoff from windrow composting pad (Tollner & Das, 2004). Consequently, a good hydrologic design and compost pad management are required as pond sizing and the runoff water quality are highly regulated (Bhattacharjee & Tollner, 2016). For the state of Gerogia, USA, composting is an important option of interest as there are forty to fifty windrow composting systems in Georgia. Management of the composting pad is crucial since the pad runoff is highly regulated and researchers tried to estimate runoff in order to provide guidance for retention pond design (Kalaba *et al.*, 2007, Wilson *et al.*, 2004).

### Data Collection

Figure 1.1 represents the study site of Bioconversion center, University of Georgia, Athens, USA. The runoff data was collected on a 10-minute interval during 11:50AM, December 23, 2010 to 11:50PM, January 30, 2011 over $T = 5445$ time points). The raw data ($g_0$) are represented by the noisy red curve (the target response) in Figure 1.2. This plot shows a few random computer

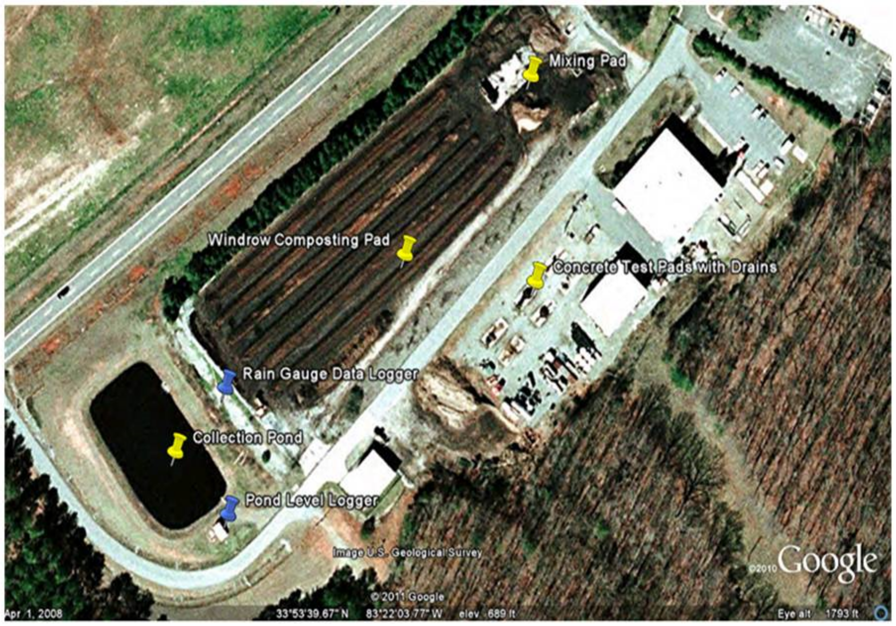


Figure 1.1: Google Earth (© 2010) overhead view of the windrow composting pad at the Bioconversion Center, UGA, Athens (Duncan *et al.*, 2013b).

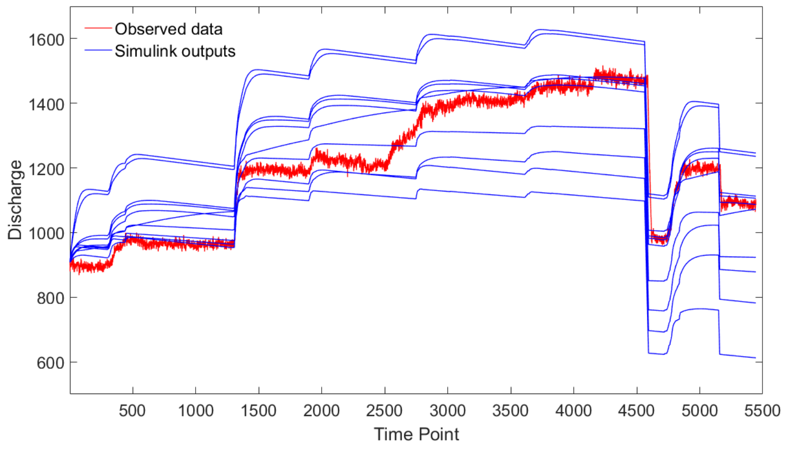


Figure 1.2: Discharge field data ($g_0(t)$) from Bioconversion center, UGA (represented by the red curve) and the Matlab-Simulink model discharge outputs $g(x,t)$ (represented by the blue lines) for $t = 1, 2, \ldots, 5445$ at randomly generated $x$ (depth of surface, depth of sub-surface and two coefficients of the saturated hydraulic conductivity $K_{sat1}$ and $K_{sat2}$ ).

model responses superimposed with the field data. The measurements were taken using rain gauge data logger which was placed between the compost pad and collection pond. The pond stage measurements were recorded using two pressure transducers. Full description of data collection and characteristics of composting pad can be found in (Duncan *et al.*, 2013b, Duncan *et al.*, 2013a).

## Model Description

Duncan et al. investigated the rainfall-runoff relationship for the windrow composting pad, and developed a compartmental Matlab-Simulink model for estimating the amount of runoff from the composting pad (represented as a change in pond volume). Figure 1.3 demonstrates the schematic representation of the compartmental model derived from a similar tank model that was developed to calculate the rate of runoff and sediment yield in a watershed (Lee & Singh, 2005).

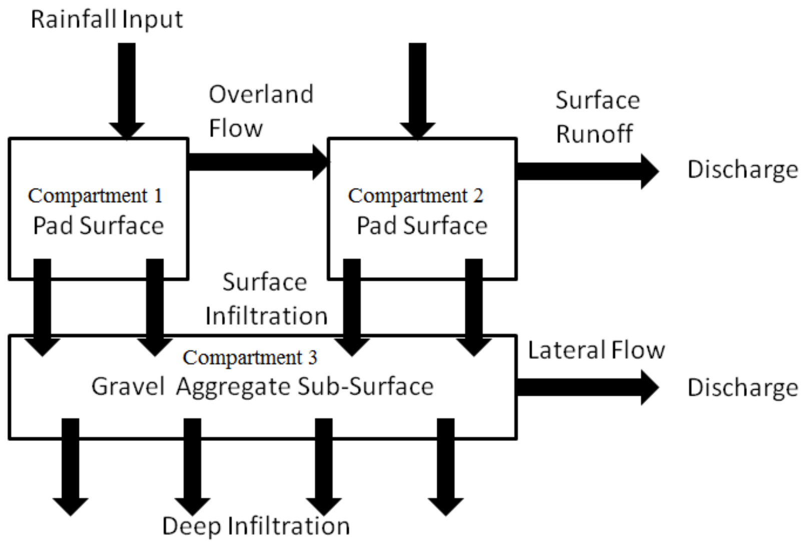


Figure 1.3: Schematic of cross-sectional view of composting pad at the Bioconversion center, UGA, Athens, USA.

The Matlab-Simulink model takes into account the effect of surface properties on the distribution of runoff and infiltration. In Figure 1.3, compartments 1 and 2 represent the up slope and down slope pad sections, respectively. The rainfall either infiltrates to the compartment 3 (gravel sub-surface) or becomes overland flow. The compartment 3 allows either deep infiltration or lateral flow. Therefore, the compartmental model quantifies the surface runoff, infiltration and lateral seepage using differential equations developed for each section of the compost pad (Duncan *et al.*, 2013b). Additionally, the model takes several factors as inputs, for instance, length, width, slope of compost pad, area covered by compost windrows, depth of surface/sub-surface, depression/embankment depths, initial surface/sub-surface water content, and model coefficients of the saturated hydraulic conductivity of the gravel media ($K_{sat1}$) and the saturated hydraulic conductivity of the supporting soil below the media ($K_{sat2}$). The detailed description of the Simulink model can be found in Duncan et al., 2013b.

It quantifies the surface runoff, infiltration and lateral seepage using differential equations developed for each section of the compost pad. Additionally, the model takes several factors as inputs, for instance, length, width, slope of compost pad, area covered by compost windrows, depth of surface/sub-surface, depression/embankment depths, initial surface/sub-surface water content, and model coefficients of the saturated hydraulic conductivity of the gravel media ($K_{sat1}$) and the saturated hydraulic conductivity of the supporting soil below the media ($K_{sat2}$). As per Duncan et al., 2013b, the following four inputs/parameters are the most influential: depth of surface, depth of sub-surface and two coefficients of the saturated hydraulic conductivity ($K_{sat1}$ and $K_{sat2}$). See Duncan et al., 2013b for more details on data collection, characteristics of composting pad and the Matlab-Simulink model.

## 1.3 Case study 2: SWAT model

### Introduction

The second case study refers to a well-known reservoir model called Soil and Water Assessment Tool (SWAT). SWAT is an internationally accepted simulator and used in modeling of the rainfall-runoff processes across various watersheds and river basins to address climate changes, water quality, land use and water resources management practices (Krysanova & Srinivasan, 2015, Dile *et al.*, 2013, Jayakrishnan *et al.*, 2005, Srinivasan *et al.*, 2005). Sequential Uncertainty Fitting (SUFI2) is one of the methods used in SWAT-CUP (Soil and Water Assessment Tool Calibration and Uncertainty Procedures) for calibration and uncertainty analysis (Abbaspour *et al.*, 2004, Abbaspour *et al.*, 2007). In this study, we used SWAT model to estimate surface runoff in the Middle Oconee River in Athens, GA. We also applied the modified history matching algorithm for calibrating SWAT model and compared the results with SUFI2.

### Data Sources

The target response was retrieved from the historical monthly data of streamflow from the US Geological Survey (USGS) water data website (gauge number 02217500) for the period of January 2001 to December 2009. Our study site is located near Middle Oconee River, Athens, USA (Figure 1.4).

For this study area, we obtained ASTER digital elevation model (DEM) values at 30m resolution from USGS EarthExplorer platform and Global Climate Data in SWAT format from Texas *A*&*M* University website. Several additional datasets were required and formatted appropriately for input to ArcSWAT within ArcMap 10.2. These included streamgage, climate, water quality, and soil datasets. Table 1.1 below summarizes the data processed for SWAT model. Figure 1.5 provides an example of SWAT input data sources such as DEM, land use data and soil data.

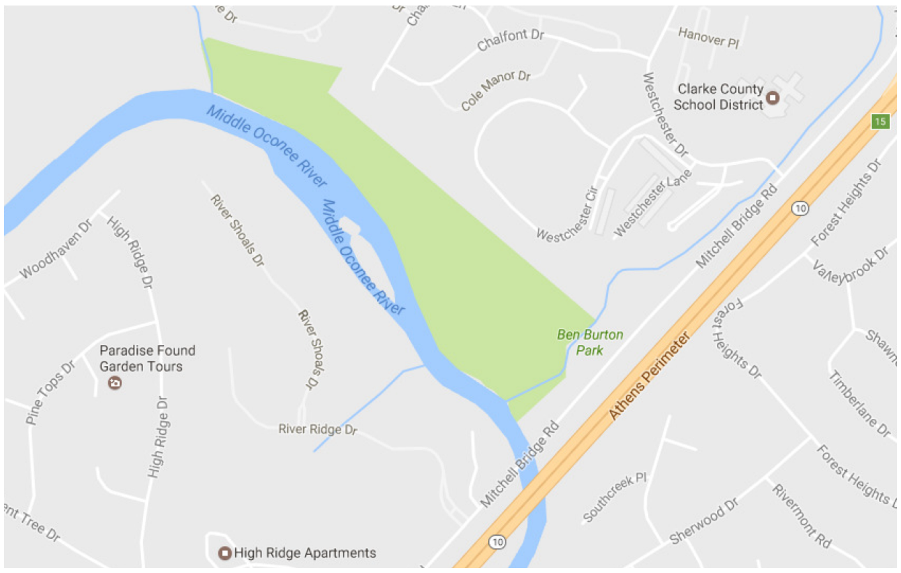


Figure 1.4: Map of the Middle Oconee River near Ben Burton Park, *Google Maps*.

Table 1.1: Summary of data processing for SWAT

| **Dataset** | **Processing** | **Data Format** |
|---|---|---|
| ASTER DEM | Reprojected to UTM 17N | DEM dataset for study area |
| USGS discharge data | Compiled for 2001-2009 | Time-series dataset for selected gauge |
| USGS gSSURGO | Reprojected to UTM 17N<br>Clipped to study area | Soil raster dataset for study area |
| SWAT Global Weather Data | Compiled for 2001-2009 | Gridded weather station data |
| NLCD 2001 | Reprojected to UTM 17N<br>Clipped to study area<br>Added ID field classes | Land cover raster dataset<br>for study area |

For USGS discharge data, we used a warm-up period of two years (January 2001 to December 2002) and a calibration period of seven years (January 2003 to December 2009). Figure 1.6 illustrates a few SWAT model runs (in blue – obtained by randomly varying the calibration inputs) and the actual field data (in red) at $T = 84$ time points.

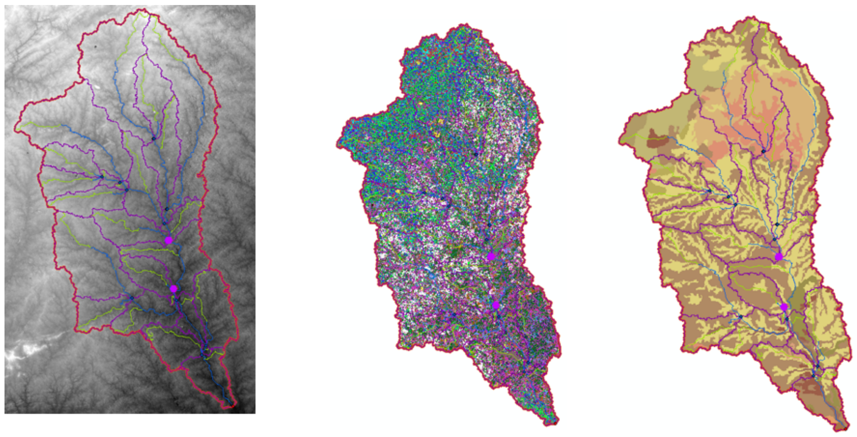


Figure 1.5: SWAT input data sources

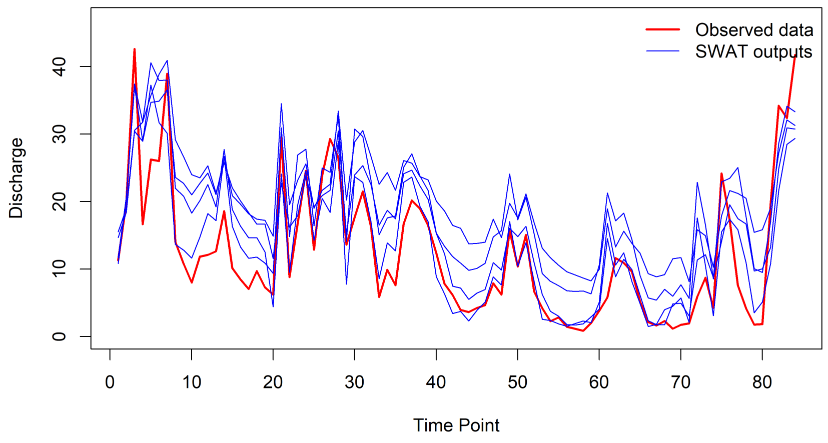


Figure 1.6: Middle Oconee river discharge field data (USGS gauge number 02217500), $g_0(t)$ (represented by the red curve), and SWAT model discharge outputs $g(x,t)$ (represented by the blue curves) for $t = 1, 2, \ldots, 84$ at randomly generated $x$ (Manning's coefficient for the main channel, effective hydraulic conductivity in the main channel, groundwater delay, groundwater "revap" coefficient and available water capacity).

## Model Description

This hydrologic model takes climate data (rainfall, air temperature, solar radiation, relative humidity and wind speed), digital elevation model (DEM) data, land use and soil type data. Figure 1.7 shows the schematics of the SWAT model we use in this paper. For SWAT modeling, we used ArcSWAT 2012.10.18 version for ArcGIS 10.2 from the Texas $A\&M$ University website.

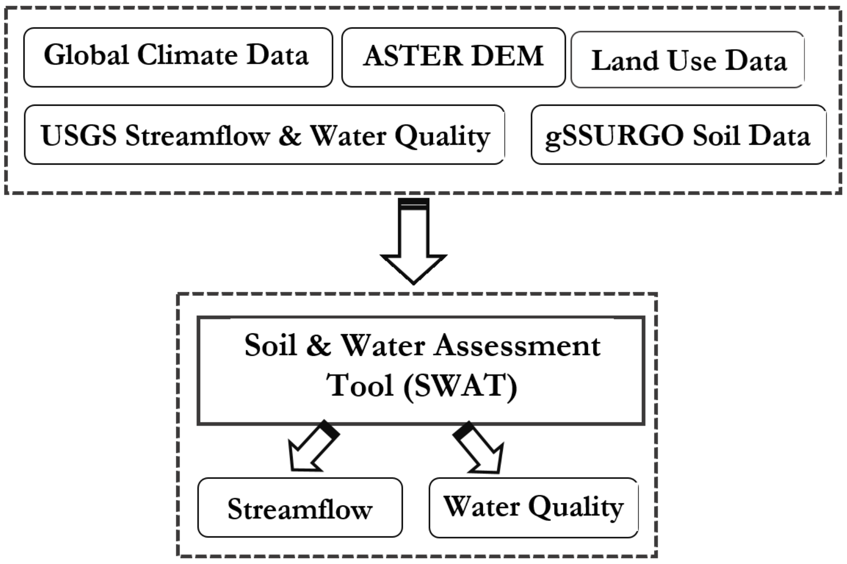


Figure 1.7: Conceptual representation of SWAT model.

There are several inputs to SWAT model, for example, curve number ($CN$), groundwater delay ($GW_{delay}$), available water capacity ($AWC$), baseflow factor ($\alpha_{BF}$), Manning's coefficient ($\nu$), etc. Based on experts' advise and preliminary variable screening analysis using Sequential Uncertainty Fitting (SUFI2) toolkit, we identified the following five parameters for the calibration exercise: $\nu$, effective hydraulic conductivity in the channel ($K$), $GW_{delay}$, groundwater "revap" coefficient ($GW_{revap}$) and $AWC$. More details on SUFI2 can be found in (Abbaspour *et al.*, 2004, Abbaspour *et al.*, 2007).

These parameters were selected from the full set based on $p\text{-}value < 0.05$, coefficient of determination $R^2 > 0.5$ and Nash-Sutcliffe efficiency ($NSE$) coefficient $> 0.5$ as the acceptable values (using SWAT CUP toolkit). The flowchart on sensitivity analysis procedure is given in Figure 1.8. The list of parameters in the full set is given in Section 3.2.

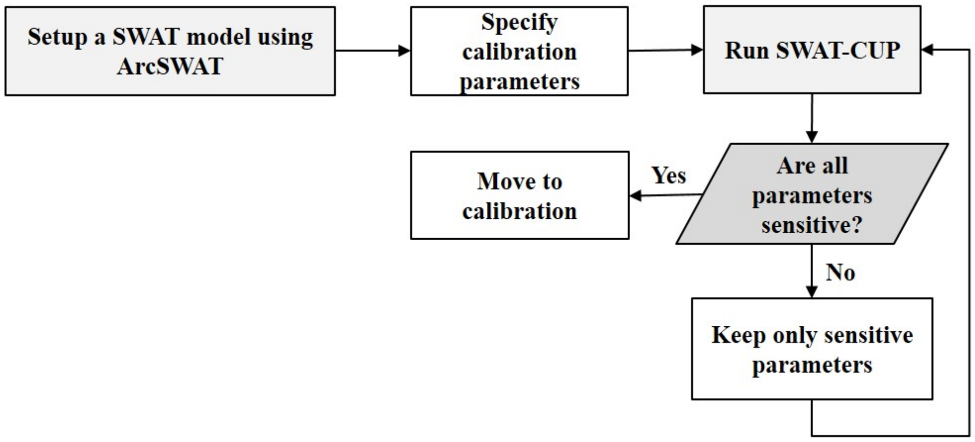

Setup a SWAT model using ArcSWAT
Specify calibration parameters
Run SWAT-CUP
Are all parameters sensitive?
Yes
Move to calibration
No
Keep only sensitive parameters

Figure 1.8: Sensitivity Analysis procedure in SWAT CUP.

## 1.4 Calibration Problem

The rainfall-runoff process is highly non-linear, time-varying and spatially distributed (Singh, 1964). The input parameters of these models are often unknown and correspond to physical properties that are difficult to measure. The input parameters of these models are high dimensional, and the outputs can be very sensitive to small changes in the inputs. Realistic computer models can also be computationally and/or financially expensive, which prohibits numerous evaluation of the simulator. As a result, the calibration of these time-series models is a challenging problem, and an efficient approach to find the inverse solution is extremely important. Several researchers have attempted to solve the inverse problem for hydrologic models using different methods via both manual and automated approaches, such as, the Genetic Algorithms, Maximum Likelihood Estimator, Markov Chain Monte Carlo, and Shuffled Complex Evolution (Duan *et al.*, 1992, Franchini & Galeati, 1997, Boyle *et al.*, 2000, Montanari & Toth, 2007, Chu *et al.*, 2010, Tigkas *et al.*, 2015).

Tuning/calibration of these parameters is required to obtain realistic outputs (Montanari & Toth, 2007). This calibration problem is also referred to as the inverse problem in computer experiments literature. We propose a modification in the HM algorithm which allows us to find the inverse solution in fewer simulator runs, and gives us a perfect match if possible, otherwise, the best approximation instead of returning an empty set of inverse solutions. We carry out a simulation study and two case studies of rainfall-runoff models to apply the proposed algorithm in solving this inverse mapping problem. To the best of our knowledge, the HM algorithms have not been applied yet for calibration of hydrologic models with time series response.

## 1.5 Summary

This research deals with obtaining the set of input parameters of a rainfall-runoff model that corresponds to a pre-specified target response, which is the observed field data. Chapter 2 provides a background of calibration problem in statistics (also referred as inverse problem). Chapter 3 focuses on a review of existing algorithms for calibration of time series model. Chapter 4 introduces the proposed approach for solving the inverse problem and includes the simulation study. Chapter 5 evaluates the performance of new approach for the two case studies and compares it with the existing algorithms. Chapter 6 gives summary and conclusion.

# Chapter 2

# Calibration Problem in Statistics

In statistics, the calibration problem is also referred to as the inverse problem in *computer experiments* literature. In this Chapter, we describe the main concept of computer experiments (Section 2.1) and provide statistical background on Gaussian Process Model (Section 2.2), Latin Hypercube Design (Section 2.3 and Expected Improvement Criterion (Section 2.4.

## 2.1 Introduction

*Computer experiments* help to analyze and understand complex physical phenomenon. Many physical processes cannot be studied experimentally due to its complexity, involved cost (in terms of time and/or money) or ethical considerations. Figure 2.1 represents illustration of the main idea behind the computer experiments − the experimentation with computer simulators.

Assuming that a realistic computer simulator of a complex physical process is also computationally and/or financially expensive, a statistical surrogate (computer emulator) is often used to emulate the simulator outputs and provide inference about physical experiments. For emulation of computer simulator, Gaussian Process model is used as a computationally inexpensive statistical tool. More detailed description on Gaussian Process model is given in Section 2.2.

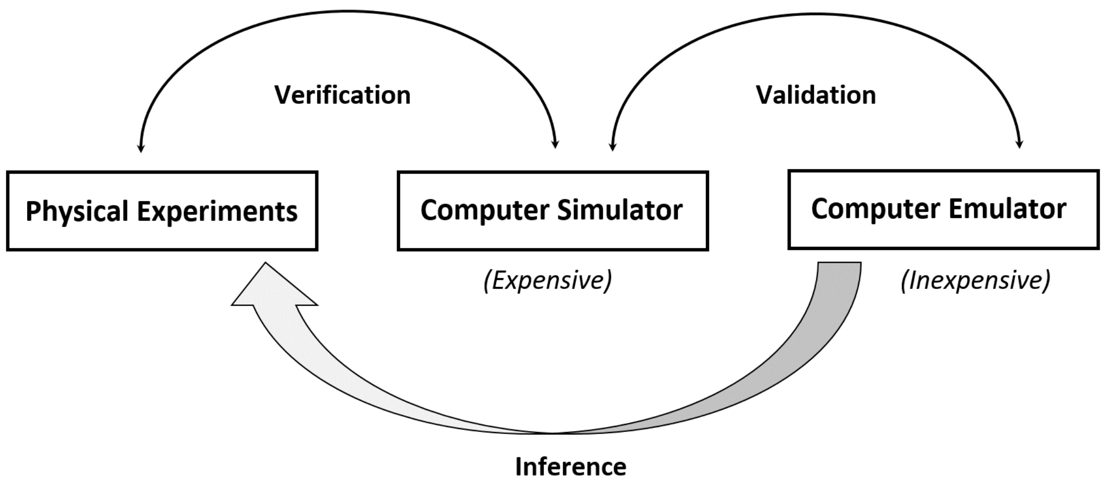


Figure 2.1: Illustration of computer experiments

Here, the time series simulator takes a $d$-dimensional vector as input and returns a time series output. Mathematically, suppose the simulator output is denoted by $g(x) := \{g(x, t), t = 1, 2, ..., T\}$ for a given input $x \in [0, 1]^d$ (scaled to an unit cube for convenience), then the objective is to find the $x$ (or set of $x$'s) that generate the desired (pre-specified) output $g_0 := \{g_0(t)$, where $t = 1, 2, ..., T\}$ (say). Thus, the computer simulator is considered to be deterministic, i.e. running the computer models with the same inputs will produce identical results (without the random error effect).

From a perspective of design of computer experiment, sampling procedure plays an important role as to what inputs would be considered when searching for an inverse problem solution. Latin Hypercube (LH) sampling was introduced as an experimental design for deterministic computer simulators (McKay *et al.*, 2000). Generating LH design is a computationally inexpensive even for a large number of input variables. Section 2.3 includes more detailed background on LH sampling.

To solve an inverse problem, we eventually need to use “data-adaptive sequential design" where we choose next run based on optimization problem (bringing $g(x)$ to $g(x_0)$ as closer as possible). Jones et al. introduced a concept of Expected Improvement (EI) to decide what the next experimental run would be (Jones *et al.*, 1998). The choice is made where EI is the largest. Formulation of EI criterion is provided in Section 2.4.

## 2.2 Gaussian Process Model

Gaussian Process (GP) is a generalization of the Gaussian probability distribution: GP provides a way of defining prior distributions over the space of continuous functions. Sacks et al. pioneered the use of a GP model (a non-parametric model) for building a surrogate for such a simulator response (Sacks *et al.*, 1989). Since then several variations have been proposed (Santner *et al.*, 2003, Rasmussen & Williams, 2006). In geology and meteorology, GP regression (known as *kriging* is widely applied in spatial modeling (Cressie, 1993). In time series, autoregressive moving average models (ARMA) and Kalman filters are known as forms of GP models (Bishop, 2006). As we mentioned in Section 2.1, GP model is the most important tool in building statistical surrogates for emulating the computer simulator.

The simplest and yet the most popular version of the GP model for $n$ training points $(x_i, y_i)$, where $i = 1, 2, ..., n$ assumes that

$$y(x_i) = \mu + Z(x_i), \quad i = 1, 2, ..., n,$$

where $\mu$ is the mean term and $Z(x_i)$ is a GP with $E(Z(x_i)) = 0$, $Var(Z(x_i)) = \sigma_z^2$ and spatial covariance structure defined as $Cov(Z(x_i), Z(x_j)) = \Sigma_{ij} = \sigma^2 R(x_i, x_j)$. Thus, $Z(\mathbf{x}) \sim GP(0, \sigma^2\mathbf{R})$ and $y(\mathbf{x}) \sim N_n(\mathbf{1}_n\mu, \sigma^2\mathbf{R})$. Instead of a constant mean $\mu$, one can use a mean function $\mu(x)$. Here, we assume that the simulator is deterministic, stationary and returns a scalar response $y(x_i)$ for every input $x_i \in [0, 1]^d$.

The most important component of the GP model, which makes it very flexible, is the correlation structure. Gaussian correlation is the most popular because of its properties like smoothness and usage in other areas like machine learning and geostatistics, whereas, both power-exponential and Matérn can be thought of as generalizations of the Gaussian correlation. The power-exponential correlation is given by

$$R(x_i, x_j) = \text{corr}(Z(x_i), Z(x_j)) = \exp\left(-\sum_{k=1}^{d} \theta_k |x_{ik} - x_{jk}|^{p_k}\right), \tag{2.1}$$

where $0 < p_k \leq 2$ are the smoothness parameters, and $\theta = (\theta_1, ..., \theta_d)$ measures the correlation lengths or the strength of long-range and short-range dependencies. Gaussian correlation corresponds to $p_k = 2$ for all $k = 1, 2, ..., d$. This model can be fitted either via the maximum likelihood estimation (MLE) or a Bayesian approach.

Under the MLE approach, the likelihood for the model is given by

$$L(\theta, \mu, \sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{\frac{1}{2}}} \times \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{1}_n\mu)^T \mathbf{R}^{-1}(y - \mathbf{1}_n\mu)\right), \tag{2.2}$$

where $\mathbf{1}_n$ is a vector of ones of length $n$, and $\Sigma = \sigma^2\mathbf{R}$. The best linear unbiased predictor (BLUP) for $y(x^*)$ at any non-sampled point $x^*$ is

$$\hat{y}(x^*) = \hat{\mu} + \mathbf{r}(\mathbf{x}^*)^T \mathbf{R}^{-1}(y - \mathbf{1}_n\hat{\mu}), \tag{2.3}$$

where $\mathbf{r}(\mathbf{x}^*) = (r_1(\mathbf{x}^*), ..., r_n(\mathbf{x}^*))^T$ and $r_i(\mathbf{x}^*) = \text{corr}(Z(x^*), Z(x_i))$ as defined in Equation 2.1. See Ranjan et al. 2008 for more details.

Under the Bayesian framework, posterior mean prediction of $y(\mathbf{x}^*)$ is

$$E[y(\mathbf{x}^*)|y_1, ..., y_n] = \mu + \mathbf{r}(\mathbf{x}^*)^T \mathbf{R}^{-1}(\mathbf{y} - \mu\mathbf{1}_n), \tag{2.4}$$

where $\mathbf{r}(\mathbf{x}^*) = [\text{corr}(z(x^*), z(x_1)), \text{corr}(z(x^*), z(x_2)), ..., \text{corr}(z(x^*), z(x_n))]^T$. The associated uncertainty (posterior variance) is

$$Var[y(\mathbf{x}^*)|y_1, ..., y_n] = \sigma^2 \left(1 - \mathbf{r}(\mathbf{x}^*)^T \mathbf{R}^{-1} \mathbf{r}(\mathbf{x}^*)\right). \tag{2.5}$$

## 2.3 Latin Hypercube Design

Latin Hypercube (LH) designs have become very popular among space-filling experimental designs for computer experiments (Iman & Conover, 1980, McKay *et al.*, 2000). They are most often used in highly dimensional problems. For example, let an experimental design with $n$ points in $k$ dimensions to be written as a $n \times k$ matrix $X = [x_1 \quad x_2 \quad \dots \quad x_p]^T$, where each column represents a variable and each row $x_i = [x_i^{(1)} \quad x_i^{(2)} \quad \dots \quad x_i^{(d)}]$ represents a sample.

Given $n \times k$ Latin hypercube $\mathbf{L} = (l_{i,j})$, a Latin hypercube design $\mathbf{D}$ in $(0, 1]^k$ design space can generated with $(i, j)^{th}$ entry being represented by Equation 2.6.

$$d_{i,j} = \frac{l_{i,j} + \frac{(n-1)}{2} + u_{ij}}{n}, \tag{2.6}$$

where $i = 1, ..., n$, $j = 1, ..., k$, $u_{ij}$'s are independent random numbers from $(0, 1]$.

Here, each of the $k$ dimensions is divided into $n$ equal levels and there is only one sample at each level. The $n$ levels are taken to be $-(n-1)/2, -(n-3)/2, ..., (n-3)/2, (n-1)/2$. This property is referred as *one-dimensional uniformity* (when the points are projected onto each axis, there is exactly one point in each of the equally-spaced intervals) (Lin & Tang, 2015). Figure 3.1 visualizes a LH in 2 dimensions.

The popularity of LH design can be explained by its capability and flexibility. This sampling technique can cover small and large design spaces without constraints in terms of data density and location (Viana, 2013). From a theoretical viewpoint, LH design is popular due to the variance reduction in numerical integration. McKay et al. (2000) established the following theorem:

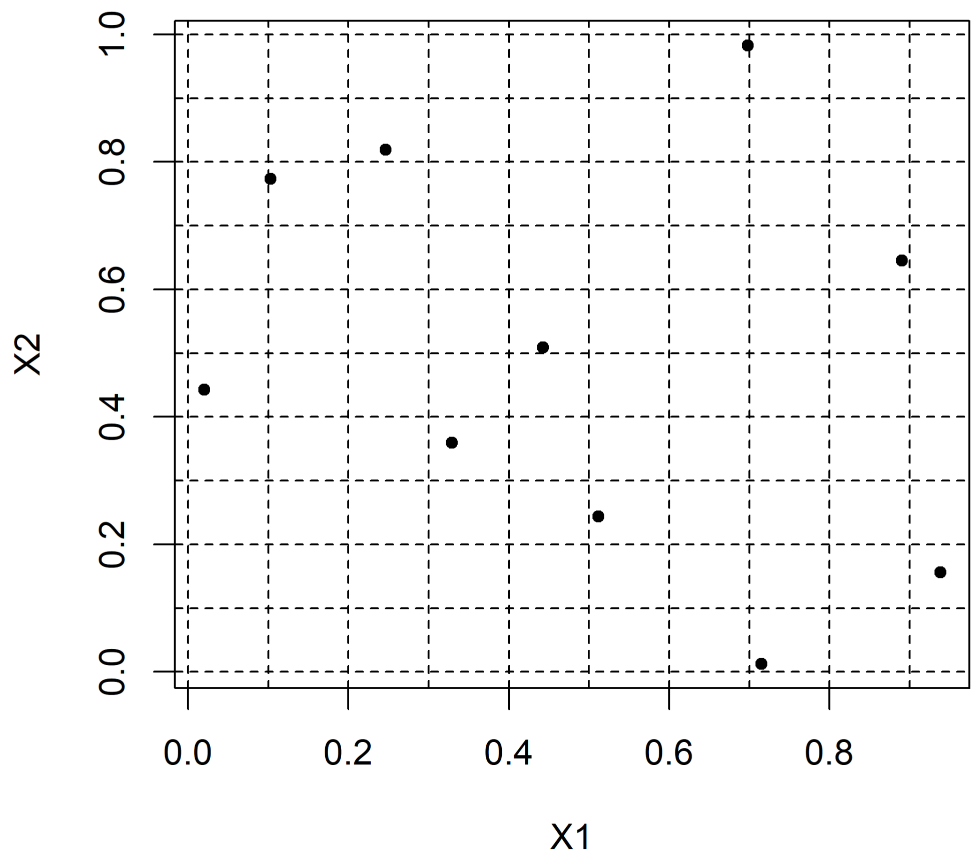


Figure 2.2: Illustration of Latin Hypercube Design

If $y = f(x)$ is monotonic in each of its input variables, then

$$Var(\hat{\mu}_{lhs}) \leq Var(\hat{\mu}_{lhs}), \tag{2.7}$$

where $\hat{\mu}_{lhs}$ is an estimate of $\mu = E(y)$ under LH sampling; and $\hat{\mu}_{lhs}$ is an estimate of $\mu$ under simple random sampling.

LH designs with space-filling properties such as maximin (Morris & Mitchell, 1995), minimum pairwise coordinate correlation, orthogonal arrays are commonly applied in computer experiments. For example, maximin distance criterion was introduced for maximizing the smallest distance between two points (to avoid points being too close) (Johnson *et al.*, 1990).

A lot of research efforts have been focused towards expanding the capabilities of Latin hypercube design, especially for cases when the first set of points was used to create a surrogate model which did not perform well. Several authors demonstrated that sequentially sampling scheme would outperform the naive LH design, a one-shot design approach (Ranjan *et al.*, 2008, Xiong *et al.*, 2009, Crombecq *et al.*, 2011, Pronzato & Müller, 2012).

## 2.4 Expected Improvement Criterion

According to Jones et al., BLUP does not account for the model uncertainty and, thus, should not be used for finding a global optimum of a process $y(x)$ (Jones *et al.*, 1998). Instead, they suggest to use "Expected Improvement (EI)" which balances local and global search.

Here, the improvement function $I(x)$ is the improvement estimate of the process $y(x)$ minimum. Thus, the improvement function can be written as

$$I(x) = \max\{f_{min} - y(x), 0\}, \tag{2.8}$$

where $f_{min}$ is the current best estimate of the global minimum. $I(x)$ is designed with the objective of finding global minimum. Jones et al. proposes that the uncertainty of the model is taken into account by taking the expectation of improvement function $E[I(x)]$.

Under the GP model framework, expected improvement function $E[I(x)]$ has a closed form expression under the predictive distribution, $y(x) \sim N(\hat{y}(x), s^2(x))$ as given by

$$E[I(x)] = s(x)\phi(u(x)) + (f_{min} - \hat{y}(x))\Phi(u(x)), \quad (2.9)$$

where $u(x) = (f_{min} - \hat{y}(x))/s(x)$. For choosing the next trial location (next experimental run) $x_n ew$, we used the following EI criterion , i.e.,

$$x_{new} = \underset{x \in [0,1]^d}{argmax}\; E[I(x)], \quad (2.10)$$

## 2.5 Summary

Here, we introduced the general idea behind the inverse problem and described the relationship between the physical system and the simulator. As discussed, we can present our belief about simulator's behavior by specifying an *emulator* which is a stochastic belief specification for a deterministic function. Therefore, it is much faster to evaluate the emulator than the simulator. More detailed information on Gaussian Process Model, Latin Hypercube Design and Expected Improvement Criterion can be found in (Jones *et al.*, 1998, Ranjan *et al.*, 2008, Mandal *et al.*, 2009).

# Chapter 3

# Review of Existing Algorithms

In this Chapter, we review some of the existing algorithms used to solve the inverse problem, compartment model and genetic algorithm (Section 3.1), SWAT and SUFI-2 algorithm (Section 3.2) and scalarization algorithm (Section 3.3).

## 3.1 Compartment Model

This section described compartment model and its approach to solving the inverse problem. Since the compost pad is a relatively uniform, this is a reasonably simple process model. Models such as HYDRUS-2D/3D®, Soil-Water-Assessment-Tool (SWAT), Soil-Plant-Atmosphere-Water (SPAW), Groundwater Loading Effects of Agricultural Management Systems (GLEAMS), or Erosion Productivity Impact Calculator (EPIC) provide estimates for runoff, subsurface flow and other hydraulic measures. An alternative to these models is necessary because of issues such as affordability and the reliance of these models on the curve number. Tollner and Das provide a detailed evaluation of the use of the curve number approach on a compost pad (Tollner & Das, 2004).

Lee and Singh used the compartmental model concept to examine sediment yield predictions from a variety of watersheds (Lee & Singh, 2005). The use of the custom compartmental model enables adjustment to analyze the different surface types that make up the compost pad individually rather than as a lump sum. The pad was also modeled using 2D version of the HYDRUS software to provide an alternative, complimentary view of the modeling process. The HYDRUS model offers the ability to evaluate the water flow dynamics along a 2-dimensional profile with different soil types. The output of the HYDRUS model can show the change in water content during the storm and direction of flow at any point within the soil profile (Duncan *et al.*, 2013b).

One may compute the outflow hydrograph resulting from a rainfall event using the following equation (Linsley Jr *et al.*, 1975):

$$q(t) = U(t - T) \times i(t)dt \tag{3.1}$$

This expression is a convolution of the unit hydrograph $U(t)$ with the excess rainfall hyetograph $i(t)$. The unit hydrograph has been used for many years in the study of rainfall-runoff relations (Huggins & Burney, 1982), and continues to be of active research interest.

The following set of first order equations were the basis for the compartment model:

$$A_1 \frac{dh_1}{dt} = V_{Rain} - V_{infil_1} - V_{overland_1} \tag{3.2}$$

$$A_2 \frac{dh_2}{dt} = (V_{Rain} + V_{overland_1}) - V_{infil_2} - V_{overland_2} \tag{3.3}$$

$$A_3 \frac{dh_3}{dt} = (V_{infil_1} + V_{infil_2}) - V_{lateral} - V_{deep} \tag{3.4}$$

where $A_1$ is the upslope surface of the compost pad, $A_2$ is the down slope section of the surface that collects storm water due to the porous embankment, $A_3$ is the surface area equal to that of combined first and second compartments.

Parameter estimates for the compartmental model were refined using a Genetic Algorithm

(GA) to find the combination of parameters that could best predict the pond volume. The GA is an optimization technique that searches through a search space of possible solutions using an evolutionary approach to evolve the search space in order to find the most probable solution (Motoki, 2002, Shah, 2010). The consecutive storm data from December 23, 2010, to January 30, 2011, served as the optimization set for the GA. During optimization, the goal was to ensure that the results would fall within the range of observed parameters (Duncan *et al.*, 2013b).

## 3.2 SWAT and SUFI-2 Algorithm

This section describes SWAT model and Sequential Uncertainty Fitting (SUFI-2) Algorithm implemented in SWAT-CUP for calibrating SWAT model.

Runoff events can be managed by protecting existing natural features and following land use best practices. The development of green infrastructure has proven to be a resourceful and cost-effective tool to address water quality and runoff management issues in city landscapes (Tzoulas *et al.*, 2007). Green infrastructure refers to a network of open space, forests, wildlife habitat, parks and other natural areas within urban and suburban areas, which help sustain clean air, water, and other natural resources (McMahon & Benedict, 2000). Green infrastructure has been shown to benefit watershed health by decreasing the effects of pollution into waterways. Specifically, urban forests were found to decrease stormwater runoff by allowing water to infiltrate and the soil to absorb particles and contaminants before entering the surface water.

The SWAT model was used to assess the impact of different land management practices, pollutions sources, and contaminants on local watersheds. We used ArcSWAT 2012.10.18 version for ArcGIS 10.2 from the Texas AM University website (ArcSWAT. Soil and Water Assessment Tool). The primary SWAT output of interest for this work was the estimated runoff for the watershed. All SWAT outputs were calibrated and validated using discharge data from USGS streamgages (USGS Current Water Data for the Nation). Traditionally, for calibration and validation purposes, researchers use SWAT CUP software that can be downloaded from Texas AM University website (SWAT-CUP. Soil and Water Assessment Tool). We used Nash-Sutcliffe Model Efficiency (NSE) as a measure of goodness-of-fit in order to evaluate SWAT model performance. Negative value of NSE means that the observed mean is a better predictor than the model. The closer NSE to 1, the better. Usually, NSE = 0.75 is considered as a good adjustment of the model (Rocha *et al.*, 2012). In previous studies, researchers tried to quantify the effect of land use/land cover on runoff and sediment yield using remote sensing, GIS and SWAT (Santhi *et al.*, 2006, Mishra *et al.*, 2007, Xu *et al.*, 2009, Tibebe & Bewket, 2011).

The SUFI-2 sequential uncertainty fitting procedure is described in more detail in Abbaspour et al. 2004. Here, we present the main steps of the calibration steps in SUFI-2:

1. In this first step an objective function is defined.

2. The second step establishes physically meaningful absolute minimum and maximum ranges for the optimized parameters.

3. Sensitivity analysis. Keeping all parameters constant to realistic values, while vary ing each parameter within the range.

4. A Latin hypercube sampling results in n parameter combinations, where n is the number of desired simulations (McKay *et al.*, 2000).

5. In this step, first sensitivity matrix is calculated $J$. Afterwards, the Hessian matrix $H$ is calculated. Finally, parameter covariance matrix $C$ is calculated.

In summary, SUFI-2 uses Latin hypercube, along with a global search algorithm that examines the behavior of an objective function by analyzing the Jacobian and Hessian matrices. The full list of parameters is listed in the Table below:

Table 3.1: Full set of SWAT parameters

| **Parameter** | **Name** | **min** | **max** |
|---|---|---|---|
| CN2 | Curve Number | -0.2 | 0.2 |
| SFTMP | Snow fall temperature | -5 | 5 |
| SURLAG | Surface runoff lag coefficient | 0.1 | 24 |
| SMTMP | Maximum Canopy Storage (mmH2O) | -5 | 5 |
| TIMP | Snow pack temp lag factor | 0 | 1 |
| ESCO | Soil evaporation compensation factor | 0.001 | 1 |
| EPCO | Plant intake compensation | 0 | 1 |
| SMFMX | Melt factor for snow in June 21 (mm H20)/C/day | 1.7 | 8 |
| SMFMN | Melt factor for snow in December 21 (mm H20)/C/day | 1.7 | 8 |
| CH(N2) | Manning's n for the main channel | 0 | 0.3 |
| CHN(1) | Manning's n for the tributary channels | 0 | 0.5 |
| CHK(2) | Effective hydraulic conductivity in the main channel alluvium (mm/hr) | 0 | 130 |
| CHK(1) | Effective hydraulic conductivity (mm/hr) | 0 | 300 |
| ALPHA-BF | Baseflow alpha factor (1/days) | 0 | 1 |
| GW_DELAY | Threshold depth of water in the shallow aquifer for revap or percolation (mm H20) | 0 | 500 |
| GWQMN | Threshold depth of water in the shallow aquifer to return flow to occur (mm H20) | 0 | 1000 |
| GW_REVAP | Groundwater ÂŞrevapÂŤ coefficient | 0 | 0.2 |
| GW_SPYLD | Specific yield of the shallow aquifer | 0 | 0.4 |
| RCHRG_DP | Deep aquifer percolation fraction | 0 | 0.2 |
| REVAPMN | Threshold depth of water in the shallow aquifer for revap or percolation (mm H20) | 0 | 500 |
| ALPHA_BNK | Base Flow alpha factor for bank storage | 0 | 1 |
| ALPHA-BF | Baseflow alpha factor (1/days) | -0.1 | 0.1 |
| SOL_AWC | Available water capacity in the soil layer (mm H20/mm soil) | -0.2 | 0.4 |
| SOL_K | Saturated Hydraulic conductivity (mm/Hr) | -0.8 | 0.8 |
| SOL_BD | Moist bulk density (mg/m3) | -0.5 | 0.6 |
| SNOCOVMX | Minimum snow water content | 0 | 500 |

## 3.3 Scalarization Algorithm

This section describes scalarization algorithm. The main idea of the so-called scalarization method is to transform this inverse problem into a minimization problem, i.e., to find the minimizer of

$$d(x) = \|g(x) - g_0\| = \left[\sum_{t=1}^{T} |g(x,t) - g_0(t)|^2\right]^{1/2}.$$

Here, we again assume that the simulator under consideration generates time-series response $g(x) = \{g(x,t), t = 1, 2, ..., T\}$, for any input $x$. Recall our objective is to find the set (or sets) of $x$'s that gives the perfect match or the best approximation of $g_0$. If the target response $g_0$ is indeed a realization of the simulator output, a perfect match can be found, otherwise, the minimization approach would lead to an approximation. Instead of using a Euclidean distance between $g(x)$ and $g_0$, one could also use Mahalanobis distance, or some other discrepancy measure more suitable from a time-series standpoint. On a cautionary note, a biased or inaccurate simulator may lead to undesirable minimizer, and the computational cost of evaluating $y(x)$ is the same as that of $g(x)$.

Since only a limited number (say $N$) of evaluations of $g(x)$ (and consequently, $y(x)$) is allowed, this minimization problem is much trickier than the standard global optimization in mathematics, computer science and machine learning literature. Jones et al. developed an innovative merit-based criterion within a sequential design framework for efficiently estimating the global minimum of a deterministic simulator with scalar-valued response (Jones *et al.*, 1998). We apply this technique on the scalarized response $y(x) = \|g(x) - g_0\|$ and find the inverse solution. Figure

The key steps are as follows:

1. *Choose a space-filling design of size $n < N$ (total budget of the training set),* $\{x_1, x_2, ..., x_n\}$ *from* $\chi = [0, 1]^d$. We considered maximin Latin hypercube designs in this paper.

2. *Evaluate the simulator at the training points* $\{x_1, x_2, ..., x_n\}$. In our case, the simulator of interest is $y(\cdot) = \|g(\cdot) - g_0\|$.

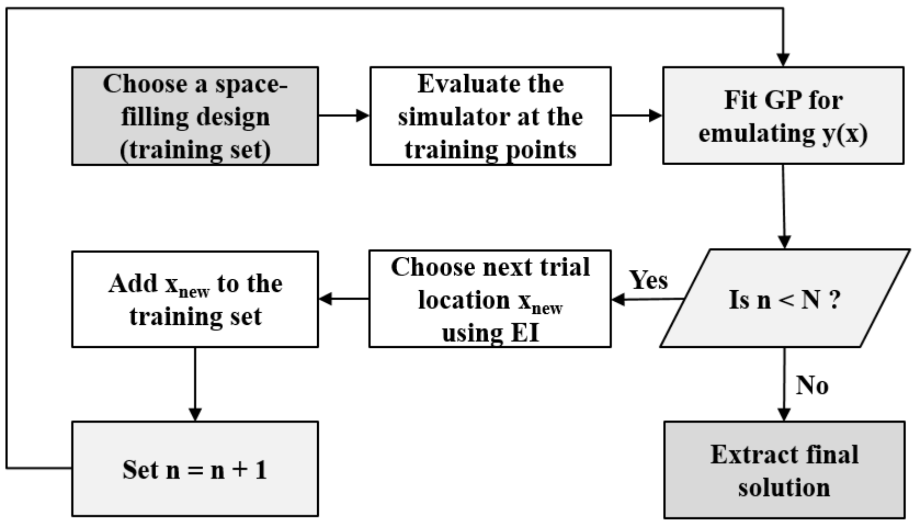

Choose a space-filling design (training set)
Evaluate the simulator at the training points
Fit GP for emulating y(x)
Is n < N ?
Yes
Choose next trial location $x_{new}$ using EI
Add $x_{new}$ to the training set
Set n = n + 1
No
Extract final solution

Figure 3.1: Flowchart of key steps in scalarization method

3. *Fit an appropriate statistical surrogate to* $\{(x_i, y(x_i)), i = 1, 2, ..., n\}$. As in Ranjan et al., we investigated the use of GP model for emulating $y(x)$ (Ranjan *et al.*, 2016).

4. *If* $n < N$*, go to Step 5, otherwise go to Step 7.*

5. *Evaluate the infill-criterion on a large test set of size* $M$.

   Similar to Jones et al. we used the expected improvement (EI) criterion for choosing the next trial location (Jones *et al.*, 1998), i.e.,

$$x_{new} = \underset{x \in [0,1]^d}{argmax}\ E[I(x)],$$

   where $E[I(x)]$ is the improvement function given in Equation 2.8.

6. *Augment the training set as* $X = X \cup \{x_{new}\}$ *and* $Y = Y \cup \{y(x_{new})\}$. *Set* $n = n + 1$ *and go back to Step 3.*

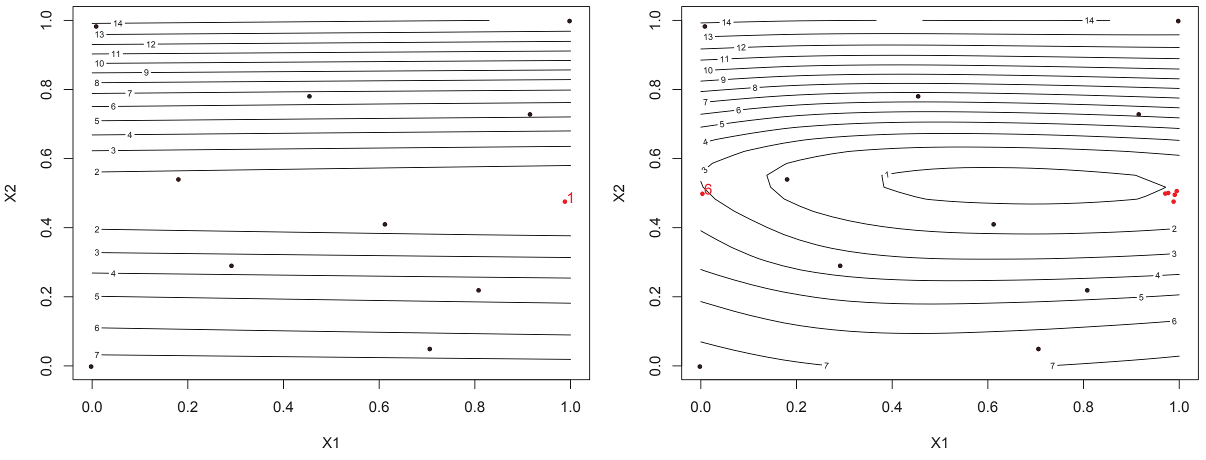
7. *Extract the final solution – the minimizer of* $\hat{y}(x)$. This is the desired inverse solution, or the best approximation we can get based on the model and data.

The step-by-step algorithm outlined here can be further generalized to accommodate a bigger class of simulators, or device a more efficient algorithm as needed by the application at hand. For instance, in Step 1, one can use another space-filling design scheme, e.g., uniform designs, orthogonal arrays, minimum coordinate correlation design, etc. (see Santner et al. 2003 for an overview). To our knowledge, there is no golden rule of selecting the right fraction for $n/N$, as the ideal choice may vary with the complexity of the simulator and the input dimension, it is common to take $N/4 \leq n \leq N/2$.

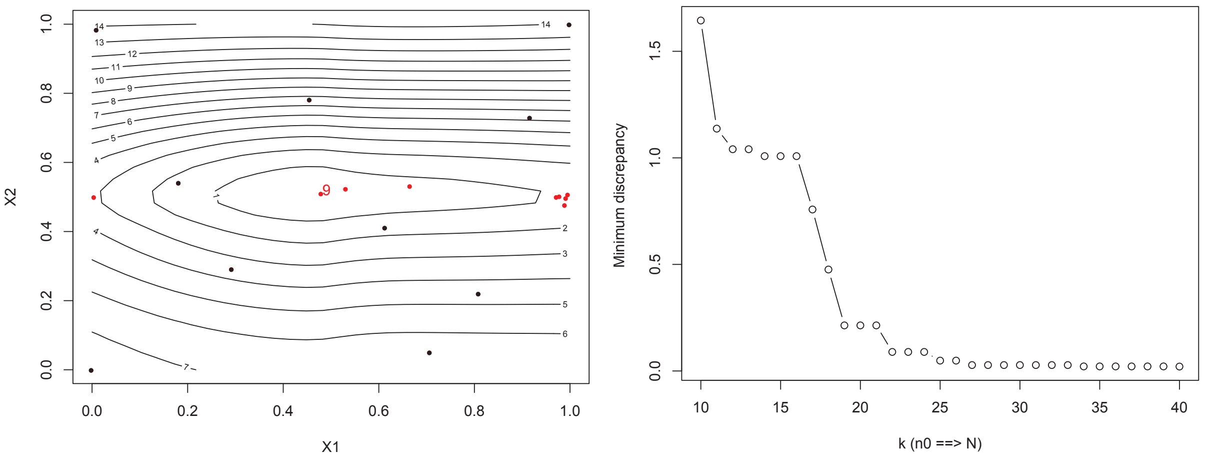
In Step 3, if any of the assumptions regarding the simulator (e.g., deterministic, stationarity, etc.) are violated, the predictive distribution and hence the functional form of the EI criterion in Step 5 may change (Picheny *et al.*, 2013, Ranjan, 2013).

Figure 3.2 presents an illustration of the scalarization method with the initial design size, $n_0 = 10$, total budget $N = 40$ and the size of the test set $M = 5000$, where the scalarized simulator $y(x) = \|g(x) - g_0\|$ is emulated via the GP model.

Figure 3.2 shows the contour plots of $\hat{y}(x)$ fitted to $n$ training points with $n_0$ initial design points and $n - n_0$ follow-up points chosen sequentially as per the EI criterion. Figure 3.2(a) shows the surface after adding one follow-up point, (b) and (c) show the estimated surface after 6 and 9 points have been added, respectively. Finally Figure 3.2(d) plots the estimated minimized $y(x)$. Figure 3.2 demonstrates that the scalarization method works reasonably well and leads to a good approximation of the global minimum of $y(x)$.


(a) $y(x)$ surface after adding 1 point

(b) $y(x)$ surface after adding 6 points


(c) $y(x)$ surface after adding 9 points

(d) Running estimate of the minimum $y(x)$

Figure 3.2: Calibration of the illustrative example using the scalarization method with $n_0 = 10$, $N = 40$, $M = 5000$ and the GP model as the surrogate.

# Chapter 4

# Proposed History Matching Approach

This Chapter is mainly focused on the proposed history matching approach by providing the details of the algorithm and its performance using simulation study results, including results on sensitivity of algorithmic parameters.

## 4.1 Introduction

Let $g(x) := \{g(x,t), t = 1, 2, ..., T\}$ denote the time series valued simulator response for a given input $x \in [0, 1]^d$ (scaled to an unit hypercube for convenience). Then the objective of the inverse problem is to find the $x$ (or set of $x$'s) that generate the desired (pre-specified) output $g_0 := \{g_0(t), t = 1, 2, ..., T\}$ (say). For many complex phenomena, the realistic computer models are also computationally and/or financially expensive to run. As a result, standard mathematical techniques and algorithms cannot be used for solving the inverse problems. Ranjan et al. proposed a sequential design approach for efficiently finding the inverse problem for scalar-valued simulators (Ranjan *et al.*, 2008). However, for this research, the complexity due to time-series response makes the problem more challenging. Ranjan et al. proposed a sequential design strategy for estimating the inverse solution, and Vernon et al. proposed an iterative approach called history matching (HM) for calibrating a galaxy formation model called GALFORM (Ranjan *et al.*, 2016, Vernon *et al.*, 2010).

## 4.2 History Matching Algorithm

The history matching algorithm proposed by Vernon et al. begins by discretizing the time-series response on $T_k$ time points, say, at $t_1^*, t_2^*, ..., t_{T_k}^*$, such that $T_k$ is much smaller than $T$ (Vernon *et al.*, 2010). These $T_k$ time points are chosen in such a way that they capture the defining features of the target response. Then, the HM method finds a common set of plausible solutions to these $T_k$ inverse problems for scalar-valued simulators, and declares it as a solution to the general inverse problem. Mathematically, the HM algorithm finds $x \in [0, 1]^d$ such that $g(x, t_j^*) = g_0(t_j^*)$ for all $j = 1, 2, ..., T_k$.

Assuming the computer model is expensive, the inverse solution must be estimated using the minimal number of model runs. A common practice in computer experiments literature is to build up the methodologies using a flexible statistical surrogate trained on carefully chosen model runs. Vernon et al. used the most popular surrogate, Gaussian process (GP) model. For simplicity, let us assume that $y(x_i) = g(x_i, t_j^*)$. As discussed in Section 2.2, the $n$ training points, $(x_i, y(x_i)), i = 1, 2, ..., n$, are modelled as $y(x_i) = \mu + Z(x_i)$, where $\mu$ is the mean and $Z(x)$ is a GP, denoted by $Z(x) \sim GP(0, \sigma^2 R)$. This implies that $E(Z(x)) = 0$ and spatial covariance structure defined as $Cov(Z(x_i), Z(x_j)) = \Sigma_{ij} = \sigma^2 R(\theta; x_i, x_j)$.

For any given input $x^*$ in the design space, the fitted GP surrogate gives the predicted simulator response as,

$$\hat{y}(x^*) = \mu + \mathbf{r}(x^*)^T \mathbf{R}^{-1}(\mathbf{y} - \mu \mathbf{1}_n), \tag{4.1}$$

where $\mathbf{r}(x^*) = [\text{corr}(z(x^*), z(x_1)), \text{corr}(z(x^*), z(x_2)), ..., \text{corr}(z(x^*), z(x_n))]^T$, $\mathbf{1}_n$ is a vector of ones of length $n$, $\mathbf{R}$ is the $n \times n$ correlation matrix for $(Z(x_1), ..., Z(x_n))$, $\mathbf{y}$ is the response vector

$(y(x_1), ..., y(x_n))$, and the associated uncertainty estimate is,

$$s^2(x^*) = \sigma^2 \left(1 - \mathbf{r}(x^*)^T \mathbf{R}^{-1} \mathbf{r}(x^*)\right). \tag{4.2}$$

In practice, the parameters $\mu, \sigma^2$ and $\theta$ in Equations (4.1) and (4.2) are replaced by their estimates (see Vernon et al. for details).

To search for input values for which $g(x) \approx g(x_0)$, we use a tolerance level defined as *Implausibility Measure*. Implausibility function is the driving force behind the HM algorithm

$$I_{(j)}(x) = \frac{|\hat{g}(x, t_j^*) - g_0(t_j^*)|}{s_{t_j}(x)}, \tag{4.3}$$

where $\hat{g}(x, t_j^*)$ is the predicted response in Equation (4.1), and $s_{t_j}(x)$ is the associated uncertainty estimate in Equation (4.2). The main idea is to label the design points implausible if $I_{max}(x) > c$, where

$$I_{max}(x) = \max\{I_{(1)}(x), I_{(2)}(x), ..., I_{(T_k)}(x)\},$$

and $c$ is a pre-determined cutoff (e.g., $c = 3$ as per $3\sigma$ rule of thumb). Vernon et al. further proposed an iterative approach to refine the plausible subset of points from the input space. However, the algorithm is designed to find the set of all plausible inverse solutions and not only the perfect solution. For the Galaxy formation model (GALFORM) application with input dimension $d = 17$, Vernon et al. used a large training set to start with ($n_1 = 1000$) and ended up with $N = 2011$ points after four iterations.

As described earlier, HM algorithm intelligently eliminates the implausible points from the input (or parameter) space and returns a set of plausible candidates for the inverse solution. However, there are a few aspects of the HM algorithm by Vernon et al. that differ from our objective. First, the end result of the HM algorithm may be an empty set if there does not exist a plausible inverse solution, and second, the HM algorithm requires a large number of simulator runs which is undesirable in several applications like ours, where the simulator is expensive to evaluate. Thus, we propose a few modifications in the history matching algorithm described in.

## 4.3 Modified History Matching Algorithm

We aim to find only the best possible approximation of the inverse solution instead of the entire plausible set, and prefer to use a reasonably small space-filling design instead of a large design in $[0,1]^d$ for building the surrogate. For instance, the rule of thumb (Loeppky *et al.*, 2006) suggests that 10 points per input dimension should be enough for getting an overall idea of the underlying process (i.e., $n_1 = 10d$, where $d$ is the input dimension). Admittedly, the rule of thumb is not full-proof, and the sufficient number of points for a good fit of the surrogate may depend on the smoothness of the underlying process.

The key steps of the proposed modified HM algorithm are summarized as follows:

1. *Choose a discretization-point-set (DPS),* $t_1^*, t_2^*, ..., t_{T_k}^*$.
2. *Set* $i = 1$. *Assume* $D_0 = \phi$ *(empty set).*
3. *Choose a training set,* $D_1 = \{x_1, x_2, ..., x_{n_1}\} \subset [0,1]^d$, *using a space-filling design, and evaluate the simulator* $g(x)$ *over* $D_1$.
4. *Fit* $T_k$ *scalar-response GP-based surrogate to* $g(x, t_j^*)$ *over the training set* $D = D_i \cup D_{i-1}$.
5. *Evaluate the implausibility criteria* $I_{(j)}(x)$ *for* $j = 1, 2, ..., T_k$ *over a randomly generated test set* $\chi_i$ *of size M (via a space-filling design) in* $[0,1]^d$ *and combine them via*

$$I_{max}(x) = \max\{I_{(1)}(x), I_{(2)}(x), ..., I_{(T_k)}(x)\},$$

   *for screening the plausible set of points* $D_{i+1} = \{x \in \chi_i \, : \, I_{max}(x) \leq c\}$.

6. *Stop if $D_{i+1} = \phi$, otherwise, set $i = i + 1$, evaluate the simulator on $D_i$ and go to Step 4.*

Instead of using the entire $D_{i+1}$ from Step 5 to Step 6, one can use a space-filling design to find a representative subset of $D_{i+1}$ and then augment it in Step 4 for the next iteration. This will further reduce the total computer model evaluation in solving the inverse problem. Since we assume that the target response is a realization of the simulator output, one can find the best possible approximation of the inverse solutions via the Euclidean distance approach, i.e., the point $x \in [0, 1]^d$ with the smallest

$$d(x) = \left( \sum_{j=1}^{T_k} |\hat{g}(x, t_j^*) - g_0(t_j^*)|^2 \right)^{1/2} \tag{4.4}$$

based on the final fit can be a decent approximation. Alternatively, one can use the minimizer of $d'(x_i) = \|g(x_i) - g_0\|$ over $x_i \in D$ as a good approximation of the desired inverse solution.

In summary, we need to identify following elements to implement the history matching algorithm:

(a) *a computer model* that takes a $d$-dimensional input vector and returns a time-series output,

(b) *input parameters* that need to be calibrated,

(c) *a target response* for calibrating the computer model, and

(d) *algorithmic parameters*: $n_1, c, T_k, (t_1^*, ..., t_{T_k}^*)$ and $M$.

Next, we present a simulation study for a comprehensive understanding of the calibration problem and investigate different aspects of the proposed algorithm (Section 4.3). Additionally, we solve the inverse problem in two real-life case studies using the modified HM algorithm (presented in Chapter 3).

## Simulation Study

The objective of this simulation study is to discuss the implementation details of the proposed algorithm, and investigate the sensitivity of the algorithmic parameters on the performance efficiency. We consider a simple test function as a computer simulator with two calibration parameters. Specifically, the inputs are $x = (x_1, x_2) \in [0, 1]^2$, which return the following time-series output:

$$g(x, t) = \frac{\sin(10\pi t)}{(2x_1 + 1)t} + |t - 1|^{(4x_2+2)}, \tag{4.5}$$

where $t = 0.5, 0.52, 0.54, ..., 2.50$ (equidistant time points of length $T = 101$). We further assume that the true value of the calibration parameter is $x_0 = (0.5, 0.5)$, which generates the target response $g_0$ in the inverse problem context. Figure 4.1 presents the model outputs for a few random input combinations (dashed curves) and the target response series (solid curve).

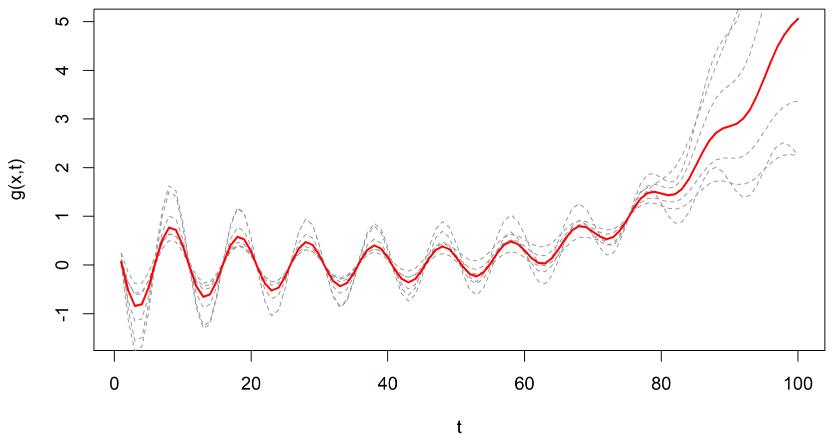
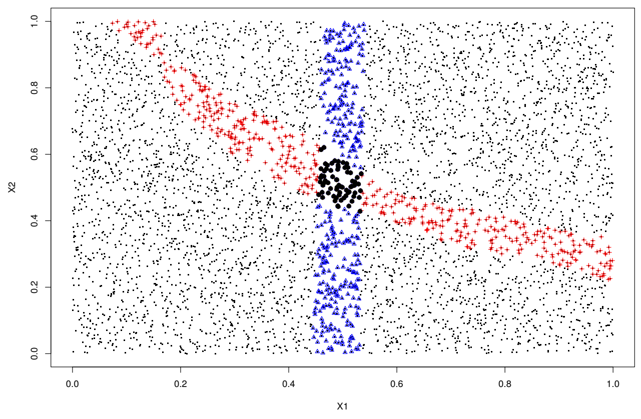


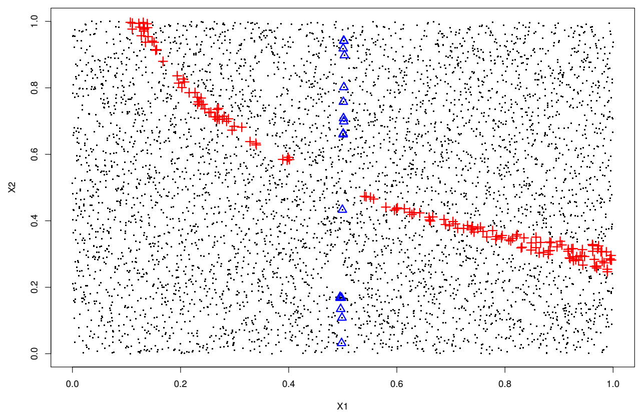
Figure 4.1: The illustrative example: a few model outputs (dashed curves) and the target response (solid curve).

Our objective is to find $x \in [0, 1]^2$ such that $g(x) \approx g_0$. We now apply the proposed HM algorithm for solving the inverse problem.

Recall that the length of the response series for this simulator is $T = 101$ and the input dimension is $d = 2$. Figure 4.2 illustrates the implementation of the algorithm with $n_1 = 10$, $c = 3$, $T_k = 2$, $DPS = (33, 67)$ and $M = 5000$.



(a) Iteration 1



(b) Iteration 2

Figure 4.2: The illustrative example: selection of the training points according to the implausibility function at the discretization-point-set $DPS = (33, 67)$ in the modified HM algorithm.

Figure 4.2(a) provides the selection of points in the first iteration, where the points in (blue) triangle and (red) plus correspond to $I_{(j)}(x) \leq 3$ for $t_1^* = 33$ and $t_2^* = 67$ respectively, and the (black) solid circle represents $D_2 = \{I_{max}(x) \leq 3\}$. Figure 4.2(b) shows the implausibility value of the candidate points in the second iteration. Given that $D_3$ (the black solid dots) is an empty set, the iterative procedure terminates.

The iterative procedure gives $n_2 = |D_2| = 69$ (i.e., the total training set size is $N = 79$), and the minimized $\log[d'(x_i)]$ over the training set as $-4.2290$, with the estimated inverse solution $(0.4992, 0.5007)$.

## Sensitivity of Algorithmic Parameters

We now investigate the sensitivity of $n_1, c, T_k, DPS$ and $M$ with respect to the test-function based computer simulator in Equation (4.5). The minimized $\log[d'(x_i)]$ over the training set is used as the goodness-of-fit measure for performance comparison. That is, the lower the value of $\log[d'(x_i)]$, the better the parameter combination is. The results are averaged over 100 random realizations. We randomly regenerated the initial training sets, test sets and the DPS for each combination of $n_1 = (5, 10, 20)$, $c = (1, 2, 3)$, $T_k = (2, 4, 8)$ and $M = (500, 2000, 5000)$, and ran the modified HM algorithm.

Figure 4.3 presents the marginal distribution of the median of the minimized $\log[d'(x_i)]$ for all possible two-factor combinations of $n_1, c, T_k$ and $M$. Here, each panel has three sub-panels. For Panel (a), the left most sub-panel corresponds to $n_1 = 5$ and the three dots there correspond to $M = 500$ (solid circle), $M = 2000$ (solid triangle), and $M = 5000$ (plus), respectively. Similarly, the middle sub-panel shows the different values of $\log[d'(x_i)]$ for three different values of $M$ and a fixed value of $n_1$(=10). The line segments in other panels and sub-panels can be explained similarly.

From Figure 4.3 we can draw some inference regarding the sensitivity and preference for the algorithmic parameters. For example, Panels (a), (b) and (c) show that as the value of $M$ increases, from 500 to 5000, the value of $\log[d'(x_i)]$ decreases monotonically. Naturally, here $M = 5000$ is the best choice. Although it may not be obvious from Panel (a), Panels (d) and (e) clearly demonstrate that $n_1 = 10$ give better results for this example, since in all of these cases, the value of $\log[d'(x_i)]$ for $n_1 = 10$ is smaller than that of $n_1 = 5$ or 20. Similarly, Panels (b) and (d) support the choice of $c = 3$, and the same conclusion can be drawn from Panel (f), since each of the three lines of this panel has the lowest value of $\log[d'(x_i)]$ at $c = 3$. Finally, Panels (c), (e) and (f), all clearly indicate

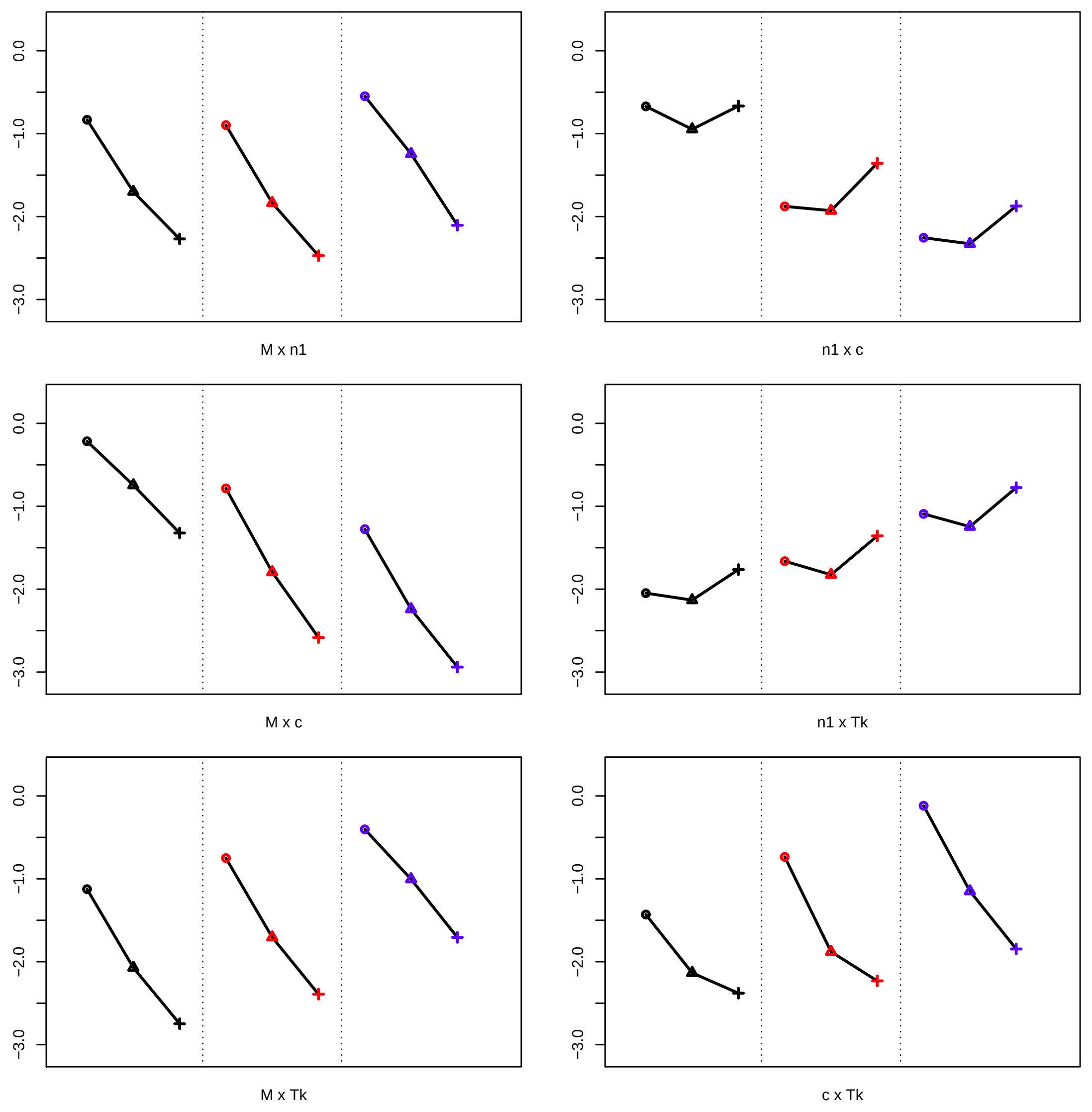


Figure 4.3: The illustrative example: marginal distribution of the median of the minimized $\log[d'(x_i)]$ over 100 simulations for different two-factor combinations of $n_1, c, T_k$ and $M$.

that $T_k = 2$ gives the lower value of $\log[d'(x_i)]$ than that for 4 and 8.

Together, these six panels of Figure 4.3 lead to some intuitive conclusions, such as the higher the value of $M$ or $c$, the better the performance of the proposed HM algorithm. However, some other conclusions are not that intuitive, and these simulations shed more light on the optimal choice of the algorithmic parameters. For example, it turns out that a higher number of dicretized points ($T_k$) may not necessarily yield a better performance of the HM algorithm. Finally, if the size of the initial design is too small or too large, the HM algorithm will not be very efficient.

As we have seen that the size of the discretization-point-set (value of $T_k$) plays a crucial role in the performance of HM algorithm, the actual location of the discretization points will also most likely affect the performance of the proposed algorithm. Figure 4.4 presents the performance comparison of the proposed algorithm over 100 simulations. We fix $n_1 = 10, c = 3$ and $T_k = 2$, and randomly generate training data and implement the algorithm under two scenarios: *Fixed* – DPS=(33, 67), and *Variable* – randomly generate DPS of size $T_k$ using some space-filling criterion. As before, the performance is measured by the optimized $\log[d'(x_i)]$ and $N$ – the total number of computer model evaluations required by the procedure.

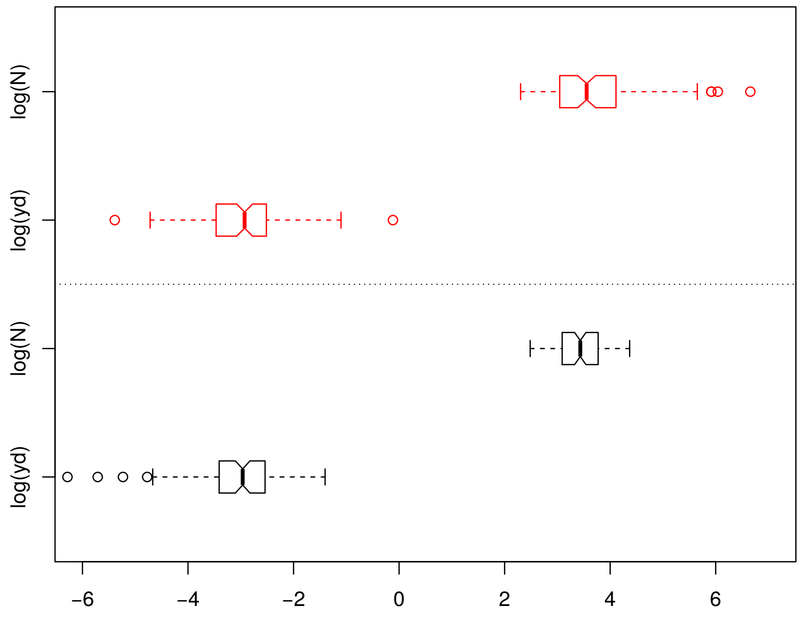


Figure 4.4: The illustrative example: Sensitivity of selecting DPS measured with respect to the total run-size and optimized $\log[d'(x_i)]$.

It is clear from the top panel of Figure 4.4 that the choice of DPS fixed at (33, 67) is clearly better than many other alternatives in terms of the total number of computer model evaluations. The bottom panel shows that both scenarios *Fixed* and *Variable* give comparable accuracy of the final inverse solution, which is expected as the termination of the algorithm depends on the accuracy of the predictor near the target response, as captured by the implausibility function in Equation (4.3). In summary, it is crucial to identify a good DPS for efficient implementation of the proposed algorithm.

## 4.4 Summary

Based on our empirical findings via a simulation study, we infer that the size of the test-set gives a trade-off between large training-set and accuracy of the inverse solution. Due to the stochastic nature of the HM algorithm, a multi-start approach of the proposed HM algorithm may lead to improved accuracy, and subsampling of $D_i$ in Step 5 may lead to more economical sampling strategy, however one must analyze the tradeoff between the accuracy gain and the additional cost of simulator evaluation for the application at hand. The choice of discretization-point-set is subjective and a key to the success of this algorithm. In practice, one should examine the target response carefully, and choose the points in such a way that they capture the overall variation and important features reasonably well.

Note that the proposed HM approach will find the closest possible approximation in case the simulator turns out to be stochastic and cannot generate the exact same desired output $g_0$. Although, it is methodologically straightforward to generalize the proposed technique that can adjust for some systematic discrepancies, a bias correction step would require synchronised data on the simulator and actual field trials for multiple input combinations.

Here, we used the Euclidean distance, $d(x) = ||g(x) - g_0||$, to sort through the test set for finding the inverse solution. Alternatively, one could use more sophisticated discrepancy measures, e.g., expected $L_2$ distance with respect to the predictive distribution, for this search.

# Chapter 5

# Performance of the Proposed Algorithm

In this chapter, we focus on calibrating the two time-series valued hydrologic models that simulate rainfall-runoff dynamics. Section 5.1 presents results on modified history matching algorithm and compartment model performance comparison (Case study 1). Section 5.2 provides results on solving inverse problem using the proposed method and quantifies SWAT model performance comparison. Data summary and model description for both case studies were provided in Chapter 1. In Section 5.3, we summarize the results.

## 5.1 Case study 1: Matlab-Simulink Model Calibration

The objective here is to find the best possible combinations of those four inputs / parameters: depth of surface, depth of sub-surface, $K_{sat1}$ and $K_{sat2}$, that can generate realistic runoff, i.e., similar to the one obtained from the field data. For convenience in the implementation of the algorithm, the inputs were scaled to $[0, 1]^4$. Following the $10d$ rule of thumb, we start the algorithm with choosing $n_1 = 40$ points using a maximin Latin hypercube design (Johnson *et al.*, 1990), and evaluate the simulator on these 40 points. By carefully examining the nature of the field data, five time points ($T_k = 5$) given by $\{135, 554, 1243, 3232, 4500\}$ were selected from the runoff series (of length $T = 5445$) to discretize the time-series responses. Furthermore, we used the test set of size $M = 5000$ and $c = 3$ for computing the implausibility values. The full implementation required $N = 461$ simulator runs to converge.

The final inverse solution obtained via the proposed HM algorithm is presented in Figure 5.1. For a benchmark comparison, we also present the best inverse solution found by Duncan et al., 2013b.

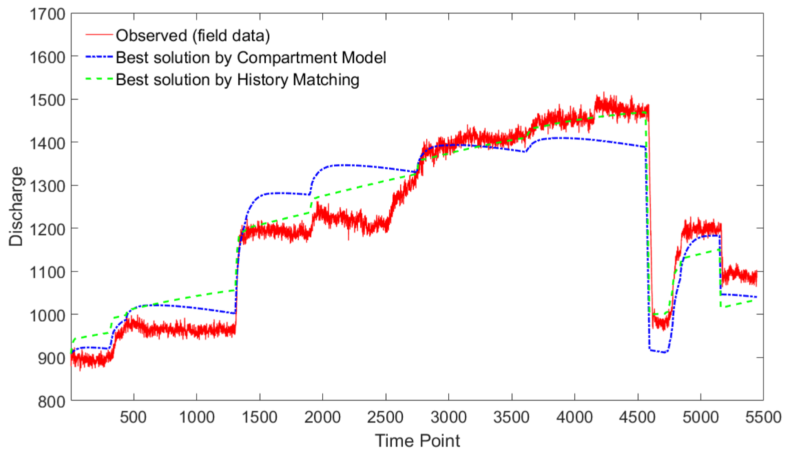


Figure 5.1: Calibration Results for the Matlab-Simulink model. The solid red curve represents observed data, blue dash line represents best solution used in the previous study and green dash line corresponds to the best solution using the proposed history matching algorithm.

Figure 5.1 shows that the proposed HM approach leads to a closer approximation of the target. Table 5.1 gives more in-depth comparison of the two approaches measured in terms of root mean squared error (RMSE), Nash-Sutcliffe Efficiency (NSE) and $R^2$.

Table 5.1: NSE, $R^2$ and RMSE comparisons for Matlab-Simulink Model

| Matlab-Simulink | NSE | $R^2$ | RMSE |
|---|---|---|---|
| Compartment | 0.86 | 0.86 | 71.71 |
| History Matching | 0.92 | 0.93 | 55.58 |

As per Table 5.1, the proposed HM algorithm outperforms the earlier approach by Duncan et al. with respect to all three goodness of fit measures, and in particular by a significant $(71.91 - 55.58)/55.58 \times 100 \approx 30\%$ margin according to RMSE.

## 5.2 Case study 2: SWAT Model Calibration

Following the steps of the proposed HM algorithm (Sect. 4.3), we rescaled the inputs to $[0, 1]^5$, assigned $n_1 = 10d = 50$ for training the initial surrogate, and carefully identified four time instances $t_j^*$ at: 10, 43, 57, 79 for discretizing the output series. Here also we used test sets of size $M = 5000$. Ultimately, the algorithm required $N = 387$ model runs to find the best solution. Figure 5.2 presents the estimated inverse solution along with the target response. For reference comparison, the best solution obtained by SUFI2 has also been overlayed in Figure 5.2.

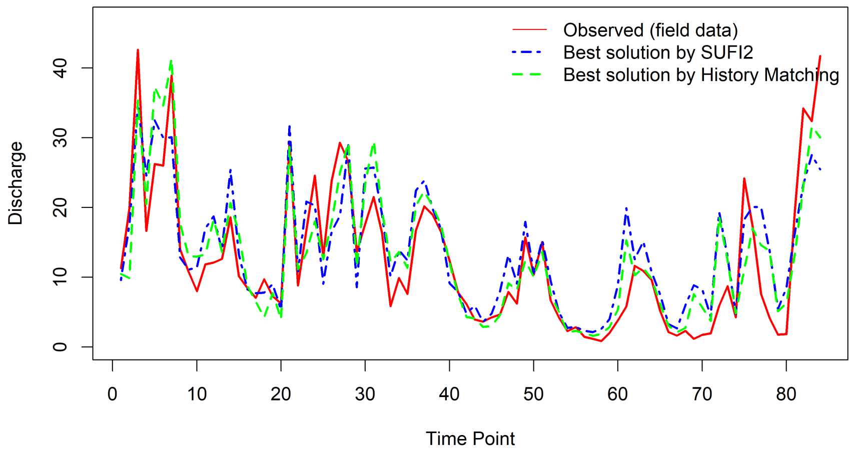


Figure 5.2: SWAT model calibration: The solid red curve represents the observed data, blue dashed line represents best solution using SUFI2 and green dashed line corresponds to the best solution using the HM algorithm.

Table 5.2 presents a more detailed comparison of the two approaches measured with respect to RMSE, NSE and $R^2$. Unlike the first case, the proposed approach did not exhibit superior performance in terms of NSE and $R^2$, but for RMSE, the proposed approach demonstrates $(5.21 - 4.70)/4.70 \times 100 \approx 11\%$ improvement over SUFI2 results (obtained from SWAT CUP 5.1.6.2 version).

Table 5.2: NSE, $R^2$ and RMSE comparisons for SWAT model

| SWAT model | NSE | $R^2$ | RMSE |
|---|---|---|---|
| SUFI2 | 0.72 | 0.74 | 5.21 |
| History Matching | 0.77 | 0.78 | 4.70 |

## 5.3 Summary

In this study, we applied the proposed modified history matching (HM) algorithm for solving an inverse problem (i.e. calibration problem) for a test function based computer model and two real-life hydrologic models. The proposed algorithm demonstrated very good performance in all scenarios. In the first case study (Matlab-Simulink model), the HM algorithm demonstrated approximately 30% better performance than the state-of-the-art compartment model calibration results. For the second case study, we observed that the HM algorithm resulted in approximately 11% more accurate inverse solution as compared to the one obtained from SUFI2. Thus, we believe that the proposed HM algorithm can be fruitful for solving calibration problems in hydrologic time-series models.

# References

Abbaspour, KC, Johnson, CA, & Van Genuchten, MT. 2004. Estimating uncertain flow and transport parameters using a sequential uncertainty fitting procedure. *Vadose Zone Journal*, **3**(4), 1340–1352.

Abbaspour, KC, Yang, J, Maximov, I, Siber, R, Bogner, K, Mieleitner, J, Zobrist, J, & Srinivasan, R. 2007. Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT. *Journal of hydrology*, **333**(2), 413–430.

Arnold, JG, Williams, JR, Srinivasan, R, King, KW, & Griggs, RH. 1994. SWAT: Soil and water assessment tool. *US Department of Agriculture, Agricultural Research Service, Grassland, Soil and Water Research Laboratory, Temple, TX.*

Bhattacharjee, NV, & Tollner, EW. 2016. Improving management of windrow composting systems by modeling runoff water quality dynamics using recurrent neural network. *Ecological Modelling*, **339**, 68–76.

Bishop, CM. 2006. *Pattern recognition and machine learning*. Springer.

Boyle, DP, Gupta, HV, & Sorooshian, S. 2000. Toward improved calibration of hydrologic models: Combining the strengths of manual and automatic methods. *Water Resources Research*, **36**(12), 3663–3674.

Chu, W, Gao, X, & Sorooshian, S. 2010. Improving the shuffled complex evolution scheme for optimization of complex nonlinear hydrological systems: Application to the calibration of the Sacramento soil-moisture accounting model. *Water Resources Research*, **46**(9).

Cressie, N. 1993. Statistics for spatial data. *Probability and Mathematical Statistics*.

Crombecq, Karel, Laermans, Eric, & Dhaene, Tom. 2011. Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *European Journal of Operational Research*, **214**(3), 683–696.

Dile, YT, Berndtsson, R, & Setegn, SG. 2013. Hydrological response to climate change for gilgel abay river, in the lake tana basin-upper blue Nile basin of Ethiopia. *PloS one*, **8**(10), e79296.

Dorahy, CG, Pirie, AD, McMaster, I, Muirhead, L, Pengelly, P, Chan, KY, Jackson, M, & Barchia, IM. 2009. Environmental Risk Assessment of Compost Prepared from Salvinia, Egeria densa, and Alligator Weed. *Journal of environmental quality*, **38**(4), 1483–1492.

Duan, Q, Sorooshian, S, & Gupta, V. 1992. Effective and efficient global optimization for conceptual rainfall-runoff models. *Water resources research*, **28**(4), 1015–1031.

Duncan, OJ, Tollner, EW, Ssegane, H, & McCutcheon, SC. 2013a. Curve Number Approaches to Estimate Drainage from a Yard Waste Windrow Composting Pad. *Applied Engineering in Agriculture*, **29**(2), 201–208.

Duncan, OJ, Tollner, EW, & Ssegane, H. 2013b. An Instantaneous Unit Hydrograph for Estimating Runoff from Windrow Composting Pads. *Applied Engineering in Agriculture*, **29**(2), 209–223.

Franchini, M, & Galeati, G. 1997. Comparing several genetic algorithm schemes for the calibration of conceptual rainfall-runoff models. *Hydrological Sciences Journal*, **42**(3), 357–379.

Huggins, LF, & Burney, JR. 1982. Surface runoff, storage and routing. *Hydrologic modeling of small watersheds*, 169–225.

Iman, RL, & Conover, WJ. 1980. Small sample sensitivity analysis techniques for computer models. with an application to risk assessment. *Communications in statistics-theory and methods*, **9**(17), 1749–1842.

Jayakrishnan, RSRS, Srinivasan, R, Santhi, C, & Arnold, JG. 2005. Advances in the application of the SWAT model for water resources management. *Hydrological processes*, **19**(3), 749–762.

Johnson, Mark E, Moore, Leslie M, & Ylvisaker, Donald. 1990. Minimax and maximin distance designs. *Journal of statistical planning and inference*, **26**(2), 131–148.

Jones, DR, Schonlau, M, & Welch, WJ. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, **13**(4), 455–492.

Kalaba, L, Wilson, BG, & Haralampides, K. 2007. A Storm Water Runoff Model For Open Windrow Composting Sites. *Compost Science and Utilization*, **15**(3), 142–150.

Krysanova, V, & Srinivasan, R. 2015. Assessment of climate and land use change impacts with SWAT. *Regional Environmental Change*, **15**(3), 431.

Lee, YH, & Singh, VP. 2005. Tank model for sediment yield. *Water Resources Management*, **19**(4), 349–362.

Lin, CD, & Tang, B. 2015. *Handbook of Design and Analysis of Experiments*. Chapman and Hall/CRC. Chap. Latin hypercubes and space-filling designs, pages 593–625.

Linsley Jr, RK, Kohler, MA, & Paulhus, JLH. 1975. Hydrology for engineers.

Loeppky, J, Bingham, D, & Welch, W. 2006. Computer model calibration or tuning in practice. *University of British Columbia, Vancouver, BC, CA*.

Mandal, A, Ranjan, P, & Wu, CFJ. 2009. G-SELC: Optimization by sequential elimination of level combinations using genetic algorithms and Gaussian processes. *The Annals of Applied Statistics*, **3**(1), 398–421.

McKay, MD, Beckman, RJ, & Conover, WJ. 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **42**(1), 55–61.

McMahon, Edward T, & Benedict, MA. 2000. Green infrastructure. *Planning Commissioners Journal*, **37**(4), 4–7.

Mishra, A, Kar, S, & Singh, VP. 2007. Prioritizing structural management by quantifying the effect of land use and land cover on watershed runoff and sediment yield. *Water Resources Management*, **21**(11), 1899–1913.

Montanari, A, & Toth, E. 2007. Calibration of hydrological models in the spectral domain: An opportunity for scarcely gauged basins? *Water Resources Research*, **43**(5).

Morris, MD, & Mitchell, TJ. 1995. Exploratory designs for computer experiments. *Journal of Statistical Planning and Inference*, **43**, 381–402.

Motoki, T. 2002. Calculating the expected loss of diversity of selection schemes. *Evolutionary computation*, **10**(4), 397–422.

Picheny, V, Ginsbourger, D, Richet, Y, & Caplin, G. 2013. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, **55**(1), 2–13.

Pronzato, L, & Müller, WG. 2012. Design of computer experiments: space filling and beyond. *Statistics and Computing*, **22**(3), 681–701.

Ranjan, P. 2013. Comment: EI Criteria for Noisy Computer Simulators. *Technometrics*, **55**(1), 24–28.

Ranjan, P, Bingham, D, & Michailidis, G. 2008. Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, **50**(4).

Ranjan, P, Thomas, M, Teismann, H, & Mukhoti, S. 2016. Inverse Problem for a Time-Series Valued Computer Simulator via Scalarization. *Open Journal of Statistics*, **6**(03), 528.

Rasmussen, CE, & Williams, CKI. 2006. Gaussian processes for machine learning. *The MIT Press, Cambridge, MA, USA*, **38**, 715–719.

Rocha, EO, Calijuri, ML, Santiago, AF, de Assis, LC, & Alves, LGS. 2012. The contribution of conservation practices in reducing runoff, soil loss, and transport of nutrients at the watershed level. *Water resources management*, **26**(13), 3831–3852.

Sacks, J, Schiller, SB, & Welch, WJ. 1989. Designs for computer experiments. *Technometrics*, **31**(1), 41–47.

Santhi, C, Srinivasan, R, Arnold, JG, & Williams, JR. 2006. A modeling approach to evaluate the impacts of water quality management plans implemented in a watershed in Texas. *Environmental Modelling & Software*, **21**(8), 1141–1157.

Santner, TJ, Williams, BJ, & Notz, W. 2003. *The Design and Analysis of Computer Experiments*. Springer Science & Business Media.

Shah, M. 2010. A genetic algorithm approach to estimate lower bounds of the star discrepancy. *Monte Carlo Methods and Applications*, **16**(3-4), 379–398.

Singh, KP. 1964. Nonlinear instantaneous unit-hydrograph theory. *Journal of the Hydraulics Division*, **90**(2), 313–350.

Srinivasan, MS, Gérard-Marchant, P, Veith, TL, Gburek, WJ, & Steenhuis, TS. 2005. Watershed scale modeling of critical source areas of runoff generation and phosphorus transport. *JAWRA Journal of the American Water Resources Association*, **41**(2), 361–377.

Tibebe, D, & Bewket, W. 2011. Surface runoff and soil erosion estimation using the SWAT model in the Keleta watershed, Ethiopia. *Land Degradation & Development*, **22**(6), 551–564.

Tigkas, D, Christelis, V, & Tsakiris, G. 2015. The global optimisation approach for calibrating hydrological models: the case of Medbasin-D model. *Pages 10–13 of: Proceedings of the 9th World Congress of EWRA*.

Tollner, EW, & Das, KC. 2004. Predicting runoff from a yard waste windrow composting pad. *Transactions of the ASAE*, **47**(6), 1953–1961.

Tzoulas, K, Korpela, K, Venn, S, Yli-Pelkonen, V, Kaźmierczak, A, Niemela, J, & James, P. 2007. Promoting ecosystem and human health in urban areas using Green Infrastructure: A literature review. *Landscape and urban planning*, **81**(3), 167–178.

Vernon, I, Goldstein, M, & Bower, RG. 2010. Galaxy formation: a Bayesian uncertainty analysis. *Bayesian Analysis*, **5**(4), 619–669.

Viana, FAC. 2013. Things you wanted to know about the Latin hypercube design and were afraid to ask. *Pages 1–9 of: 10th World Congress on Structural and Multidisciplinary Optimization*. sn.

Wilson, BG, Haralampides, K, & Levesque, S. 2004. Stormwater runoff from open windrow composting facilities. *Journal of Environmental Engineering and Science*, **3**(6), 537–540.

Xiong, F, Xiong, Y, Chen, W, & Yang, S. 2009. Optimizing Latin hypercube design for sequential sampling of computer experiments. *Engineering Optimization*, **41**(8), 793–810.

Xu, ZX, Pang, JP, Liu, CM, & Li, JY. 2009. Assessment of runoff and sediment yield in the Miyun Reservoir catchment by using SWAT model. *Hydrological Processes*, **23**(25), 3619–3630.

# Appendices

# Appendix A

# MATLAB code for the case study

```
clear;

clc;

tic;


%############################################################
%              Step 1: Loading the data                     #
%############################################################


data=xlsread('calibration.xlsx','B2:G5446'); % Calibration Set

pond=data(:,5); % pond volume in gal

pond2=pond/(7.48*(3.28^3)); % convert pond volume from gal to m^3


Time=1:length(pond2);

Time=(Time')*10/(60*24); % 10 min interval data expressed in Days


w0 = pond2; % field data (observed discharge data)
```

```
nT = length(w0); % number of data points in w0

Tp = [135 554 1243 3232 4500]; % specify Discretization-Point-Set (DPS)

w0Tp = w0(Tp); % field data at instances of Tp points of DPS
nTp = length(Tp); % number of data points in DPS

%##############################################################
%              Step 2: Initialization                         #
%##############################################################

d = 4; % dimension of training set (4 input parameters)
n0 = 40; % number of points in training set (following the 10*d rule)
ntest = 5000; % number of points in testing set
cutoff=3; % c - cutoff for selecting plausible set of points

crn_pts = [ones(1,d)*10^(-2); ones(1,d)]; % min and max values
x = lhsdesign(n0-2,d); % generate x points using Latin Hypercube Design
Xtrain = [x;crn_pts]; % training set (40x4 dimension)

%############################################################
%          Plot Matlab-Simulink example                     #
%############################################################

h = figure(1); % create new figure window
time = 1:nT; % create "time point" variable (x-axis)

% plot the observed discharge data over time
```

```
plot(time,w0,'-r','Linewidth',1);

hold on;

% generate and plot Simulink model outputs for ten randomly generated

% x points

for i=1:10

    eval = simulator(x(i,:));

    plot(time,eval,'-b', 'Linewidth',1);

    hold on;

end


% customize the figure, add labels and legends

xlim([1 5500]); ylim([500 1700]);

hleg1 = legend('Observed data','Simulink outputs');

set(hleg1,'Location','NorthWest','FontSize',18,'Linewidth',2);

legend boxoff;

xlabel('Time Point'); ylabel('Discharge');

set(gca,'fontsize',18);


% save the figure in 'fig' format

saveas(h, 'simulink_illustration','fig');


%##############################################################

%              Step 3: Running the Simulator                 #

%##############################################################


yd = zeros(n0,1); % distance values will be stored here

ytrain_mat = zeros(n0,nTp); % simulator outputs for DPS will be stored here

n = n0; % number of points for training
```

```
for i = 1:n
    % generate Simulink model outputs for the training set
    gval = simulator(Xtrain(i,:));
    % extract the Simulink model outputs for Discretization-Point-Set (DPS)
    ytrain_mat(i,:) = gval(Tp);
    % calculate Euclidean distance weight function (returns matrix of
    % distances)
    tmp = dist([w0 gval]);
    % extract a distance value from the matrix of distances
    yd(i) = tmp(1,2);
end

%##############################################################
%#         Step 4: Fitting GP models                          #
%##############################################################

nvec = n0; % number of points for training
xn = Xtrain;  % training set (40x4 dimension)
yn_mat = ytrain_mat; %  Simulink model outputs for the training DPS

wave_continue = 1;
wave_count = 0;
GPoptions.algo=0.5;

while (wave_continue == 1)
% implausibility values for testing set of DPS will be stored here
    Ikmat = zeros(ntest,nTp);
```

```
% generate x points for testing set using Latin Hypercube Design
    Xtest = lhsdesign(ntest,d);
    wave_count = wave_count + 1;

    fprintf('wave count = %d\n',wave_count);

    for k=1:nTp
        ytrain = yn_mat(:,k);
% fit GP model
        [yhat,mse]=model_fit(xn,ytrain,Xtest,GPoptions);
% evaluate the implausibility criteria^2 for the testing DPS set
        Ik2 = ((yhat-w0Tp(k)).^2./mse).*(mse>0);
% extract the implausibility criteria values for the testing DPS set
        Ikmat(:,k)=sqrt(Ik2);

        fprintf('%d-th time point in DPS \n',k);
    end

    IM = zeros(ntest,1); % I_max will be stored here
    for i=1:ntest
% find I_max among DPS points for the testing set
        IM(i) = max(Ikmat(i,:));
    end

    fprintf('%d min_IM value \n',min(IM))

% stop if no plausible set is found
    if (min(IM)>cutoff)
```

```
        wave_continue=0;
    end

% continue to search for a plausible set
    if (min(IM)<=cutoff)
        ID = find(IM<cutoff);
        nnew = length(ID); % number of new points in a plausible set
        fprintf('%d-th evaluation \n',nnew);

        n = n+nnew; % add the new points to the training set
        nvec = [nvec,nnew];  % combined number of points

        xnew = Xtest(ID,:); % the plausible set of x design points

        ynew_mat = zeros(nnew,nTp);  % simulator outputs for the
        % plausible DPS will be stored here
        for i=1:nnew
% generate Simulink model outputs for the plausbile set
            gval = simulator(xnew(i,:));
% extract the model outputs for the plausbile DPS
            ynew_mat(i,:) = gval(Tp);
        end

% combine training set and the plausbile set of x design points
        xn = [xn;xnew];
% combine model outputs for the training DPS and the plausbile DPS
        yn_mat = [yn_mat;ynew_mat];
    end

% stopping criteria
    if (wave_count>=5)
        wave_continue=0;
    end
end

%#####################################################
%#      Step 5: Finding the inverse solutions       #
%#      via Euclidean distance approach             #
%#####################################################

ymat = zeros(n,nT); % Simulink model outputs will be stored here
yd = zeros(n,1); % Euclidean distance values will be stored here

for i=1:n
% generate Simulink model outputs for the solution candidates set
    ymat(i,:) = simulator(xn(i,:));
% calculate corresponding Euclidean distance values
    yd(i) = sqrt(sum((ymat(i,:)-w0').^2));
end

opt_ID = find(yd==min(yd));
% find the inverse solution
xopt = xn(opt_ID,:);
% find the response of the inverse solution
yd_min = yd(opt_ID);
```

```
% save the workspace
filename = 'maincode_HM_msm_rng12.mat';
save(filename)
t_toc = toc;

%##############################################################
%#     Step 6: Calculating and comparing RMSE values          #
%##############################################################

filename = 'maincode_HM_msm_rng12.mat';
load(filename)

% Best solution by History Matching
w_sol = simulator(xopt);
% RMSE of History Matching
error = rmse(w0,w_sol);

% Best solution by compartment model
w_compmod = simulator([0.06589,0.73938,0.196,0.2098]);
% RMSE of compartment model
error_compmod = rmse(w0, w_compmod);

%############################################################
%          Plot Matlab-Simulink Results                     #
%############################################################

h = figure(2);  % create new figure window
```

```
time = 1:nT;  % create "time point" variable (x-axis)


% plot the observed discharge data over time
plot(time,w0,'-r','Linewidth',1);
hold on;
% plot the best solutions by compartment model and history matching
plot(time,w_compmod,'-.b',time,w_sol,'--g','Linewidth',2);


% customize the figure, add labels and legends
xlim([1 5500]); ylim([800 1700]);
hleg1 = legend('Observed (field data)','Best solution by
        Compartment Model','Best solution by History Matching');
set(hleg1,'Location','NorthWest','FontSize',18,'Linewidth',2);
legend boxoff;
xlabel('Time Point'); ylabel('Discharge');
set(gca,'fontsize',18);


% save the figure in 'fig' format
saveas(h, 'simulink_results_rng12','fig');
```

# Appendix B

# R code for the case study

```
rm(list=ls())

library(hydroGOF)

library(lhs)

library(SAVE)

###########################################

#           Initialization                #

###########################################

d = 5; # dimension of training set (4 input parameters)

n0 = 50; # number of points in training set (following the 10*d rule)

ntest = 5000; # number of points in testing set

cutoff = 3; # c - cutoff for selecting plausible set of points

#############################################

#  Running ArcSWAT and SWAT-CUP interface   #
```

```
#############################################

##################
#  Load the data #
##################

# "Flow_at_random_x.txt" contains outputs from SWAT simulator
# for randomly generated points and observed flow discharge
Q_data = read.table("flow_at_random_x.txt", head=T)
Q_data_frame = data.frame(Q_data)

# Qo is field data (observed discharge data)
Qo = Q_data_frame$obs
# Qs1-Qs8 are model outputs for randomly generated x points
Qs1 = Q_data_frame$sim1
Qs2 = Q_data_frame$sim2
Qs3 = Q_data_frame$sim3
Qs4 = Q_data_frame$sim4
Qs5 = Q_data_frame$sim5
Qs6 = Q_data_frame$sim6
Qs7 = Q_data_frame$sim7
Qs8 = Q_data_frame$sim8

# number of data points in Qo
n = length(Qo)

Tp = c(10, 43, 57, 79) # specify Discretization-Point-Set (DPS)
QoTp = Qo[Tp] # field data at instances of Tp points of DPS
```

```
nTp = length(Tp) # number of data points in DPS

############################################################
#                   Plot SWAT example                      #
############################################################

png('swat_illustration.png', width=8, height=5, units='in', res=450)
# create "time point" variable (x-axis)
time = seq(1,n)

# plot the observed discharge data over time
plot(time, Qo, type = "l",col = "red",lwd = 2, axes = FALSE,
xlab = "Time Point", ylab = "Discharge",
    ylim = c(0,1.1*max(max(Qs1),max(Qo))))

# plot SWAT model outputs for randomly generated points
lines(Qs1, lwd=1, col='blue')
lines(Qs5, lwd=1, col='blue')
lines(Qs6, lwd=1, col='blue')
lines(Qs8, lwd=1, col='blue')

# customize the figure, add labels and legends
axis(side = 1, at = seq(0,80,10))
axis(side = 2, at = NULL)
legend("topright", legend=c("Observed data","SWAT outputs"),
col=c('red','blue'), lwd=c(2,1), bty="n")
box()
dev.off()
```

```
#############################################
#  Running ArcSWAT and SWAT-CUP interface   #
#############################################

###########################################
#           Load the results file         #
###########################################

filename = file.choose()
Q_data = read.table(filename, head=T)
Q_data_frame = data.frame(Q_data)

# field data (observed discharge data)
Qo = Q_data_frame$observed
# Best solution by SWAT CUP SUFI2
Qsim.swat = Q_data_frame$sim.swat
# Best solution by history matching
Qsim.hm = Q_data_frame$sim.hm

###########################################################
#               Plot SWAT Results                         #
###########################################################

png('swat_results.png', width=8, height=5, units='in', res=450)

#  create "time point" variable (x-axis)
time = seq(1, n)
```

```
# plot the observed discharge data over time
plot(time, Qo, type="l", col="red", lwd=2, axes = FALSE,
    xlab="Time Point", ylab="Discharge",
    ylim=c(0,1.1*max(max(Qs1),max(Qo))))

# plot the best solutions by SWAT CUP SUFI2 and history matching
lines(Qsim.swat, lty=4, col='blue', lwd=2)
lines(Qsim.hm, lty=2, lwd=2, col='green')

# customize the figure, add labels and legends
axis(side = 1, at = seq(0,80,10))
axis(side = 2, at=NULL)
legend("topright",legend=c("Observed (field data)",
    "Best solution by SUFI2", "Best solution by History Matching"),
    col=c('red','blue','green'), lwd=c(1,2,2),lty=c(1,4,2),bty="n")
box()
dev.off()

############################################################
#                Model Comparisons                          #
############################################################

##########################################################
#       Calculate and Compare Models Performance: SWAT   #
##########################################################

# SWAT CUP SUFI2 model performance
```

```
SSR1 = sum((Qsim.swat-Qo)^2)

MSE1 = SSR1/n

RMSE1 = sqrt(MSE1)

ggof(sim = Qsim.swat, obs = Qo)


# History Matching model performance

SSR2 = sum((Qsim.hm-Qo)^2)

MSE2 = SSR2/n

RMSE2 = sqrt(MSE2)

ggof(sim = Qsim.hm, obs=Qo)


##########################################################

#    Calculate and Compare Models Performance: Simulink  #

##########################################################


###########################################

#           Load the results file         #

###########################################


filename = file.choose()

Q_data = read.table(filename, head=T)

Q_data_frame = data.frame(Q_data)


# field data (observed discharge data)

Qo = Q_data_frame$observed

# Best solution by Compartment model

Qs1 = Q_data_frame$sim.compmod

# Best solution by history matching
```

```
Qs2 = Q_data_frame$sim.hm


# number of data points in Qo

n = length(Qo)


#############################################

#  Calculate Model Performance Statistics   #

#############################################


# Compartment model performance

SSR1 = sum((Qs1-Qo)^2)

MSE1 = SSR1/n

RMSE1 = sqrt(MSE1)

RMSE1

ggof(sim=Qs1, obs=Qo)


# History Matching model performance

SSR2 = sum((Qs2-Qo)^2)

MSE2 = SSR2/n

RMSE2 = sqrt(MSE2)

RMSE2

ggof(sim=Qs2, obs=Qo)
```

# Appendix C

# R code for the simulation study

```
###############################################
###############################################
#          I. Main Code HM 2d                 #
###############################################
###############################################

###############################################
#         Computer Simulator Function         #
###############################################

computer_simulator <- function(param){
    p1 = 1+param[1]*2
    p2 = 2+param[2]*4
    nT = 100
    t_vec = seq(0.5,2.5,length=nT)
    w_vec = sin(10*pi*t_vec)/(p1*t_vec)+abs((t_vec-1))^(p2)
    return(w_vec)
}

field_data <- function(param){
  p1 = 1+param[1]*2
  p2 = 2+param[2]*4

  nT=100

  t_vec = seq(0.5,2.5,length=nT)
  w_vec = sin(10*pi*(t_vec))/(p1*t_vec)+(abs(t_vec-1))^(p2)
  return(w_vec)
}

##################################################
#       True parameters and field data w0        #
##################################################

# true input parameters (p1_true and p2_true)
p1_true= 0.5
p2_true=0.5
# true field data
w0 = computer_simulator(c(p1_true,p2_true))
# number of data points in w0
nT = length(w0)

##############################################################
#    Two-dimensional inputs to computer model x=(x1,x2)     #
```

```
#############################################################

d = 2 # dimension of a training set (2 input parameters)
nres = 5
x1seq = seq(0,1,length=nres)
x2seq = seq(0,1,length=nres)
xgrid = expand.grid(x=x1seq,y=x2seq)
n = nres^d # number of points in the training set
# generate xmat points for the training set
xmat = matrix(0,n,d)
xmat[,1]=xgrid[,1]
xmat[,2]=xgrid[,2]

#############################################################
#                    Computer model output w(x)             #
#############################################################

# simulator outputs will be stored here
w_mat = matrix(0,n,nT)

# generate model outputs for the training set
for(i in 1:n){
    w_mat[i,]=computer_simulator(xmat[i,])
}

################################################################
#      Here we generate the time series plot                   #
################################################################
```

```
png('illustrative_example.png',width = 8, height = 5,
                               units = 'in', res = 450)

# create "time point" variable (x-axis)
Time = seq(1,100,1)

# plot true field data over time
plot(Time,w0,type="l",col="red",lwd=2,xlab="t",
    ylab=expression("g(x,t)"),ylim=c(-1.5,5))
# plot simulator outputs for random points
for(i in seq(1,n,4)){
    lines(Time,w_mat[i,],lty=2,col="gray60")
}
# plot true field data over time over the gray lines
lines(Time,w0,type="l",col="red",lwd=2)
dev.off()

#############################################################
#          Proposed History Matching                        #
#############################################################

w0Tp = w0[Tp]
nTp = length(Tp)
Xtrain = maximinLHS(n0,d)

ytrain_mat = matrix(0,n0,nTp)
for(i in 1:n0){
  g_vec = computer_simulator(Xtrain[i,])
  ytrain_mat[i,] = g_vec[Tp]
}

n = n0
nvec = n0
xn = Xtrain
yn_mat = ytrain_mat

wave_continue = TRUE
wave_count = 0

while(wave_continue == TRUE){
  Ikmat = matrix(0,ntest,nTp)
  Xtest = randomLHS(ntest,d)
  wave_count = wave_count + 1



  col_vec = c("blue","red","magenta","green")
  plot(Xtest,pch='.',cex=2, xlab="X1",ylab="X2")
  for(k in 1:nTp){
    fit = GP_fit(xn,yn_mat[,k],control=c(50*d,50,10))
    pred = predict(fit,xnew=Xtest)
    yhat = pred$Y_hat
    mse = pred$MSE
    Ik2 = matrix(0,ntest,1)
    Ik2 = ((yhat-w0Tp[k])^2/mse)*(mse>0)
    Ikmat[,k]=sqrt(Ik2)
    points(Xtest[(sqrt(Ik2)<=cutoff),],pch=(k+1),
                  cex=1.5,lwd=2,col=col_vec[k])
  }
  IM = apply(Ikmat,1,max,na.rm=TRUE)
  points(Xtest[(IM<=cutoff),],pch=19,col="black")

  if(min(IM)>cutoff){wave_continue=FALSE}

  if(min(IM)<=cutoff){
    ID = which(IM<cutoff)
    nnew = length(ID)

    n = n+nnew
    nvec = c(nvec,nnew)

    xnew = Xtest[ID,]
    if(nnew==1){dim(xnew)=c(1,d)}

    ynew_mat = matrix(0,nnew,nTp)
    for(i in 1:nnew){
      g_vec = computer_simulator(xnew[i,])
      ynew_mat[i,] = g_vec[Tp]
    }

    xn = rbind(xn,xnew)
    yn_mat = rbind(yn_mat,ynew_mat)
  }
  if(wave_count>=5){wave_continue=FALSE}
}

ymat = matrix(0,n,nT)
yd = matrix(0,n,1)
for(i in 1:n){
  ymat[i,] = computer_simulator(xn[i,])
  yd[i] = sqrt(sum((ymat[i,]-w0)^2))
}
xopt = xn[which.min(yd),]
yd_min = yd[which.min(yd)]

cat('History matching results\n')
cat('Run sequence = ',nvec,'\n')
print(xopt)
print(log(yd_min))
print(wave_count)

#############################################################
# generating random Latin hypercube design n points         #
#############################################################

png('lhd_design.png', width=5, height=5, units='in', res=450)
set.seed(2)
x10 = maximinLHS(10,2)

plot(x10, pch=20, xlab="X1",ylab="X2")
```

```
abline(v=seq(0,1,0.1), lty=2, lwd=1)

abline(h=seq(0,1,0.1), lty=2, lwd=1)

dev.off()


###############################################

###############################################

#           II.  Supercode HM                 #

###############################################

###############################################


d = 2

ntest_vec = c(500,2000,5000)

n0_vec = c(5,10,20)

cutoff_vec = c(1,2,3)

nTp_vec = c(2,4,8)

nsim = 100

len = 3


cat(file='output_HM_new.txt',append=FALSE)


iabhyu = 0


for(i1 in 1:len){

  ntest = ntest_vec[i1]


  for(i2 in 1:len){

    n0 = n0_vec[i2]
    for(i3 in 1:len){
      cutoff = cutoff_vec[i3]

      for(i4 in 1:len){
        nTp = nTp_vec[i4]

        for(i5 in 1:nsim){
          set.seed(i5)

          source("HM_function_new.R")
          cat(file='output_HM_new.txt',sep=',',ntest,n0,cutoff,
                                nTp,n,xd_opt,yd_opt,append=TRUE)
          cat(file='output_HM_new.txt','\n',append=TRUE)

          iabhyu = iabhyu+1; cat("Point #", iabhyu, "done on",
                              as.character(Sys.time()), "\n")

        }
      }
    }
  }
}

################################################
################################################
#           III.  Supercode HM 2d              #
################################################
################################################
```

```
nsim = 100

Tp = c(33,67);  # solve history matching on these time-points

#Tp = sort(floor(maximinLHS(2,1)*100)+1);

# solve history matching on these time-points

#set.seed(1)


cat(file='output_hm_2d_sim_seed.txt',append=FALSE)




    for(i1 in 1:nsim){

      set.seed(i1)

      source("maincode_HM_2d_sim.R")

      cat(file='output_hm_2d_sim_seed.txt',sep=',',n,yd_min,append=TRUE)

      cat(file='output_hm_2d_sim_seed.txt','\n',append=TRUE)

    }


###############################################

###############################################

#         IV.  Main code HM 2d sim            #

###############################################

###############################################


rm(list=ls())


library(lhs)

library(GPfit)
```

```
d = 2

n0 = 10

ntest = 5000

cutoff=3


#############################################################

#         Computer Simulator Function                       #

#############################################################



computer_simulator <- function(param){

  p1 = 1+param[1]*2

  p2 = 2+param[2]*4



  nT=100



  t_vec = seq(0.5,2.5,length=nT)

  w_vec = sin(10*pi*(t_vec))/(p1*t_vec)+(abs(t_vec-1))^(p2) #+ 0.5*t_vec

  return(w_vec)

}


field_data <- function(param){

  p1 = 1+param[1]*2

  p2 = 2+param[2]*4



  nT=100



  t_vec = seq(0.5,2.5,length=nT)

  w_vec = sin(10*pi*(t_vec))/(p1*t_vec)+(abs(t_vec-1))^(p2)
  return(w_vec)
}

############################################################
#                  True field data w0                      #
############################################################

p1_true=0.5; p2_true=0.5;
p = cbind(p1_true,p2_true)
w0 = field_data(c(p1_true,p2_true))
nT = length(w0)
w0Tp = w0[Tp]
nTp = length(Tp)
Xtrain = maximinLHS(n0,d)

ytrain_mat = matrix(0,n0,nTp)
for(i in 1:n0){
  g_vec = computer_simulator(Xtrain[i,])
  ytrain_mat[i,] = g_vec[Tp]
}

n = n0
nvec = n0
xn = Xtrain
yn_mat = ytrain_mat

wave_continue = TRUE
wave_count = 0
```

```
while(wave_continue == TRUE){
  Ikmat = matrix(0,ntest,nTp)
  Xtest = randomLHS(ntest,d)
  wave_count = wave_count + 1

  for(k in 1:nTp){
    fit = GP_fit(xn,yn_mat[,k],control=c(50*d,50,10))
    pred = predict(fit,xnew=Xtest)
    yhat = pred$Y_hat
    mse = pred$MSE
    Ik2 = matrix(0,ntest,1)
    Ik2 = ((yhat-w0Tp[k])^2/mse)*(mse>0)
    Ikmat[,k]=sqrt(Ik2)

  }
  IM = apply(Ikmat,1,max,na.rm=TRUE)

  if(min(IM)>cutoff){wave_continue=FALSE}

  if(min(IM)<=cutoff){
    ID = which(IM<cutoff)
    nnew = length(ID)

    n = n+nnew
    nvec = c(nvec,nnew)

    xnew = Xtest[ID,]
    if(nnew==1){dim(xnew)=c(1,d)}

    ynew_mat = matrix(0,nnew,nTp)
    for(i in 1:nnew){
      g_vec = computer_simulator(xnew[i,])
      ynew_mat[i,] = g_vec[Tp]
    }

    xn = rbind(xn,xnew)
    yn_mat = rbind(yn_mat,ynew_mat)

  }
  if(wave_count>=5){wave_continue=FALSE}
}

ymat = matrix(0,n,nT)
yd = matrix(0,n,1)
for(i in 1:n){
  ymat[i,] = computer_simulator(xn[i,])
  yd[i] = sqrt(sum((ymat[i,]-w0)^2))
}
xopt = xn[which.min(yd),]
yd_min = yd[which.min(yd)]
```