entropy

Article

Expectation-Maximization Algorithm for the Calibration of Complex Simulator Using a Gaussian Process Emulator

Yun Am Seo [1] **and Jeong-Soo Park** [2,*]

[1] AI Weather Forecast Research Team, National Institute of Meteorological Sciences, Seogwipo 697-010, Korea; seoya98@korea.kr
[2] Department of Statistics, Chonnam National University, Gwangju 61186, Korea
* Correspondence: jspark@jnu.ac.kr; Tel.: +82-62-530-3445

Abstract: The approximated non-linear least squares (ALS) tunes or calibrates the computer model by minimizing the squared error between the computer output and real observations by using an emulator such as a Gaussian process (GP) model. A potential defect of the ALS method is that the emulator is constructed once and it is no longer re-built. An iterative method is proposed in this study to address this difficulty. In the proposed method, the tuning parameters of the simulation model are calculated by the conditional expectation (E-step), whereas the GP parameters are updated by the maximum likelihood estimation (M-step). These EM-steps are alternately repeated until convergence by using both computer and experimental data. For comparative purposes, another iterative method (the max-min algorithm) and a likelihood-based method are considered. Five toy models are tested for a comparative analysis of these methods. According to the toy model study, both the variance and bias of the estimates obtained from the proposed EM algorithm are smaller than those from the existing calibration methods. Finally, the application to a nuclear fusion simulator is demonstrated.

Keywords: best linear unbiased predictor; code tuning; iterative algorithm; Latin-hypercube design; mean squared error; metamodel; numerical optimization; surrogate

Citation: Seo, Y.A.; Park, J.-S. Expectation-Maximization Algorithm for the Calibration of Complex Simulator Using a Gaussian Process Emulator. *Entropy* **2021**, *23*, 53. https://doi.org/10.3390/e23010053

Received: 29 November 2020
Accepted: 29 December 2020
Published: 31 December 2020

1. Introduction

Modern researchers have attempted to develop and use simulation code instead of excessively expensive or infeasible physical experiments in many fields. Advanced computer technology has made it possible to realize very complicated simulations. In most cases, the simulation code contains several unknown parameters (or universal constants). A classic method for determining the universal constants in computer models is non-linear least squares estimation (NLSE). It makes the sum of the squared error between the real observations and the computer responses as minimum. The NLSE, however, will become too computationally expensive or infeasible in terms of time when the computer model is time-consuming to run. In such cases, a statistical emulator can be used to determine the universal constants in the computer model so that the simulator or emulator can represent the real experiments effectively well. This process is known as "code tuning" or "calibration" [1–4].

Code tuning is defined as the procedure of enhancing the consistency between code responses and experimental data via determining the parameters inside the simulation code [5]. A differentiation can be made between tuning parameters and calibration parameters [3]. However, we use these two terms (calibration and code tuning) interchangeably.

Our work focuses on calibration in a frequentist manner [6–8], rather than a Bayesian approach [2,9,10]. A Bayesian calibration with a bias correction was introduced by Kennedy and O'Hagan [2]. Contributions to this topic include the related works of sequential tuning [11], multivariate computer outputs [12], identifiability [13,14], multi-fidelity [9], good designs for calibration [10], GP models for big data [7], and various metamodeling applications [8,15,16].

Calibration has often been performed within a framework, in the literature, in which the code suffers from a systematic bias or discrepancy for any parameter values. This shows a view that the mathematical equations inside the simulator may not perfectly describe the real world [2,12]. This framework may be more realistic, but it is beyond the range of this study. Thus, presentation in this study is focused on a statistical method that does not include the simulation discrepancy [1]. It would be, however, possible to extend our framework when the shape of the bias were available.

Cox et al. [1] studied an approximated NLSE in the form of approximated nonlinear least squares (ALS) for calibration. They utilized the Gaussian process (GP) as an emulator (or a surrogate or metamodel) of the complex simulator. The ALS first builds the GP model from the computer data. Then, it regards the built GP model as if it were true simulator. This surrogation makes the NLSE computationally feasible. We adopt the GP model as an emulator of the complex simulator, in this study. The GP model has been used successfully in constructing a regression model for complex data [17–21] and in analyzing computer experiments [22–25].

A drawback of the ALS is that the emulator (a GP model) is constructed once from the computer data only and it is not re-modeled after that. To address this defect, Seo et al. [4] considered an iterative modification of the ALS, which they referred it as the "max-min" algorithm. The max-min algorithm improves the calibration accuracy and acquires stable results. In this study, we propose another iterative method based on the expectation-maximization (EM) algorithm. We anticipate that this new method is more effective than existing approaches, including the ALS, max-min algorithm, and likelihood-based calibration. The proposed method is compared with existing calibration methods using five toy-models, in which the true values are known a priori.

This report is arranged as follows: a GP model for simulation experiments is described in Section 2. Existing methods including the ALS, max-min algorithm, and likelihood-based calibration are briefly introduced in Section 3. EM-based calibration is presented in Section 4. In Section 5, a toy model study is described. The application to a nuclear fusion simulator is presented in Section 6. Discussions are outlined in Section 7, followed by a summary in Section 8. Further details including technical specifications are provided in the Appendix. Certain contents of this study are inevitably similar to those of Reference [4,6] because the problem setting of these studies was very similar.

2. GP Model for Computer Experiments

Sacks et al. [22] proposed the adoption of the GP model for the analysis of deterministic computer experiments. The GP model can be expressed as follows, for the response $Y(\mathbf{t})$ at site $\mathbf{t}$:

$$Y(\mathbf{t}) = \sum_{i=1}^{p} \beta_i f_i(\mathbf{t}) + Z(\mathbf{t}), \tag{1}$$

where β_is are unknown regression coefficients, $\{f_i(\cdot)\}$ is a set of known functions, and $Z(\cdot)$ is the GP (random variables) with mean zero and covariances $\sigma_Z^2 R(\mathbf{t})$. In the above equation, the first term indicates a regression function and the second part ($Z(\mathbf{t})$) stands for the departure from the assumed regression function. The stochastic process term allows the predictions (5) to interpolate the deterministic responses. For $\mathbf{t}_i = \{t_{i1}, ...t_{id}\}$ and $\mathbf{t}_j = \{t_{j1}, ...t_{jd}\}$, the covariance between $Z(\mathbf{t}_i)$ and $Z(\mathbf{t}_j)$ is denoted by

$$\mathrm{Cov}(Z(\mathbf{t}_i), Z(\mathbf{t}_j)) = V(\mathbf{t}_i, \mathbf{t}_j) = \sigma_Z^2 R(\mathbf{t}_i, \mathbf{t}_j), \tag{2}$$

where σ_Z^2 is the variance of $Z(\cdot)$ and $R(\mathbf{t}_i, \mathbf{t}_j)$ is a correlation function. We select the Gaussian correlation family [26]:

$$R(\mathbf{t}_i, \mathbf{t}_j) = \exp\left(- \sum_{k=1}^{d} \theta_k (t_{ik} - t_{jk})^2 \right), \tag{3}$$

where $\theta_k's$ are non-negative parameters. We define $v'(\mathbf{t}_0)$ and $f'(\mathbf{t}_0)$ as

$$v'(\mathbf{t}_0) = [V(\mathbf{t}_0, \mathbf{t}_1), ..., V(\mathbf{t}_0, \mathbf{t}_n)], \quad f'(\mathbf{t}_0) = [f_1(\mathbf{t}_0), ..., f_p(\mathbf{t}_0)]. \tag{4}$$

In this case, $v'(\mathbf{t}_0)$ is a correlation vector between the observed (or training) sites and a prediction site $\mathbf{t}_0$, and $f'(\mathbf{t}_0)$ is a design (or functional) vector of $\mathbf{t}_0$.

When the correlation function $R(\cdot, \cdot)$ is assigned and the observations y at sites $\mathbf{t}$ are given, the best linear unbiased predictor (BLUP) of $Y(\mathbf{t}_0)$ is [26]

$$\hat{Y}(\mathbf{t}_0) = [v'(\mathbf{t}_0), f'(\mathbf{t}_0)] \begin{bmatrix} V & F \\ F' & 0 \end{bmatrix}^{-1} \begin{bmatrix} y \\ 0 \end{bmatrix} = f'(\mathbf{t}_0)\widehat{\beta} + v'(\mathbf{t}_0)V^{-1}(y - F\widehat{\beta}), \tag{5}$$

where $V = V(\mathbf{t}, \mathbf{t})$ is an $n \times n$ covariance metrix, $F = [f_j(\mathbf{t}_i)]_{1 \le i \le n, 1 \le j \le p}$ is an $n \times p$ design matrix, and $\widehat{\beta} = (F'V^{-1}F)^{-1}F'V^{-1}y$ is the GLSE (generalized least squares estimator) of β. Furthermore, the hyper-parameters in V are generally unknown, and hence, we usually estimate the hyper-parameters using the MLE (maximum likelihood estimation) from the observations y. These estimates are subsequently plugged into (5), causing (5) to become the so-called empirical BLUP (EBLUP) of $Y(\mathbf{t}_0)$ [26]. We used the "DiceKriging" [27] package of the R program. A brief description of the MLE calculation is provided in Appendix A.

We can build the prediction models differently according to the combination of $\theta's$ and $\beta's$ in (1) and (3). We construct the following two models, in this study:

$$Y(x) = \beta_0 + \beta_1 t_1 + ... + \beta_d t_d + Z(\mathbf{t}) + \varepsilon, \tag{6}$$

$$\text{Model 1: with common } \theta, \quad \text{Model 2: with } d \text{ different } \theta's.$$

In the above, "common θ" indicates that d number of $\theta's$ are constrained to be a common θ_c so that $\theta_1 = \theta_2 = \cdots = \theta_d := \theta_c$. The final error term ($\epsilon$) is for the randomness or measurement error in the real experiments. The error term is not applied to the computer responses because only a deterministic computer model is considered in this study. Other models such as the one from variable selection algorithms [28–30] are of course usable. We recommend References [26,31] for further details on the GP model.

3. Existing Calibration Methods

3.1. Data Structure and Notations

The experimental data and computer data are subscrived by "E" and "C", respectively, and the combined (both) data are denoted by "B". Let $\boldsymbol{\tau}$ be a vector of calibration parameters with the dimension q and $\underline{T}$ represent the input variables of the simulator corresponding to $\boldsymbol{\tau}$. We denote X for the experimental input variables with the dimension p. In addition, let n_C, n_E be the sample sizes and $n_B = n_C + n_E$. More details are presented in Appendix B.

3.2. ALS Method

In this subsection, we briefly describe the ALS method considered in Reference [1]. If the simulation code is computationally costly to run, it is very difficult to optimize numerically a certain quantity from the code directly in terms of time. In this case, the ALS employs the GP model as a surrogate or an emulator of the computer code. The ALS first estimates the GP parameters using the MLE for the simulation data. Thereafter, it regards the built GP model as if it were true simulator. The ALS determines $\widehat{\boldsymbol{\tau}}$ by minimizing the following approximated residual sum of squares (ARSS):

$$ARSS(\boldsymbol{\tau}) = \sum_{i=1}^{n_E} \left[y_{E,i} - \hat{Y}(\boldsymbol{\tau}, \mathbf{x}_{E,i}) \right]^2, \tag{7}$$

where $\hat{Y}(\boldsymbol{\tau}, \mathbf{x}_E)$ is the EBLUP of $Y(\mathbf{x}_0)$, as in (5).

The advantage of the ALS method is that it does not need additional runs of the simulator to calculate $ARSS(\boldsymbol{\tau})$ after a GP surrogate has been constructed from a computer dataset. Because there is no explicit minimizer in $ARSS(\boldsymbol{\tau})$, we employ the quasi-Newton iteration in the "optim" package of the R program.

A potential disadvantage of the ALS method is that the emulator (a GP model) is constructed once from the simulation data and it is no longer re-built. To address this defect, an iterative method of the ALS, namely the max-min algorithm, was considered by Seo et al. Reference [4].

3.3. Max-Min Algorithm

The tuning constants and GP parameters are alternately estimated by ALS method and the MLE, in the max-min algorithm. This method utilizes both experimental and computer data repeatedly until convergence. The steps are outlined as follows:

Step 1: Find the MLE of the GP parameters $\hat{\theta}'s$, $\widehat{\beta}'s$, and $\hat{\sigma}_Z^2$ in Equations (2) and (6) using the simulation data ($T_S = (\boldsymbol{\tau}_S, \mathbf{x}_S)$ and y_S) only.
Step 2: Determine $\widehat{\boldsymbol{\tau}}$, which minimizes the $ARSS(\boldsymbol{\tau})$ in (7) with the estimates $\hat{\theta}'s$, $\widehat{\beta}'s$, and $\hat{\sigma}_Z^2$ from Step 1.
Step 3: [maximization] Find the MLE of the GP parameters $\hat{\theta}'s$, $\widehat{\beta}'s$, and $\hat{\sigma}_Z^2$ using the combined data (T_B and y_B), where $T_B = \begin{pmatrix} T_E^* \\ T_S \end{pmatrix}, y_B = \begin{pmatrix} y_E \\ y_S \end{pmatrix}, T_E^* = \{t_{E,1}^*, ..., t_{E,n_E}^*\}'$, and $t_{E,i}^{*\,\prime} = (\widehat{\boldsymbol{\tau}}, x_{E,i})$.
Step 4: [minimization] Determine $\widehat{\boldsymbol{\tau}}$, which minimizes the $ARSS(\boldsymbol{\tau})$ in (7) with the estimates $\hat{\theta}'s$, $\widehat{\beta}'s$, and $\hat{\sigma}_Z^2$ from Step 3.
Step 5: Repeat Step 3 and Step 4 until convergence, such as $\sum_{i=1}^{q} |\widehat{\tau}_i^{old} - \widehat{\tau}_i^{new}| \,/\, |\widehat{\tau}_i^{old}| < \epsilon$.

In Steps 1 and 3, a GP model is constructed for the prediction. Steps 2 and 4 are the same in minimizing $ARSS(\boldsymbol{\tau})$, but Step 2 utilizes computer data only in the prediction $\hat{Y}(x_E)$, whereas Step 4 uses both data. In Step 3, $\widehat{\boldsymbol{\tau}}$ is the estimate obtained from the previous step. Steps 1 and 3 are the same in terms of seaching the MLE, but Step 1 utilizes computer data only, whereas Step 3 utilizes both data. Quasi-Newton numerical algorithms are employed for the optimizations in Steps 2–4.

Seo et al. [4] demonstrated the max-min method works better than the ALS. One defect of this method is that it needs more computing time than the ALS. Further details on this algorithm, including the stopping rule, can be found in [4].

3.4. Likelihood-Based Calibration

Cox et al. [1] considered the full likelihood function using the combined data for all parameters, including the calibration parameters $\boldsymbol{\tau}$; variance parameter $\sigma_{\epsilon E}^2$ and GP parameters θ, $\underline{\beta}$, and σ^2. Thus, all parameters are estimated simultaneously by the MLE method. This method is called as the full MLE, and it is applied to a GP model. The minus two times profile log likelihood function of all parameters is, without constants,

$$-2\,log\;L(\boldsymbol{\tau}, \psi_B;\, \underline{y}_B, X_B) \;=\; n_B\,log\,\hat{\sigma_B^2} \;+\; log\,|V_B|, \tag{8}$$

where

$$\hat{\sigma_B^2} = (\underline{y}_B - F_B\widehat{\underline{\beta}}_B)^t V_B^{-1} (\underline{y}_B - F_B\widehat{\underline{\beta}}_B)/n_B, \tag{9}$$

$$\widehat{\underline{\beta}}_B \;=\; ({F_B}^t V_B^{-1} F_B)^{-1} {F_B}^t V_B^{-1} \underline{y}_B, \tag{10}$$

in which $\psi_B = (\underline{\theta}_B, \underline{\beta}_B, \sigma_B^2, \gamma_E)$ with $\widehat{\underline{\beta}}_B$ and $\hat{\sigma_B^2}$ inserted, and $\gamma_E = \sigma_{\epsilon E}^2/\sigma^2$.

Cox et al. [1] considered other likelihood-based approaches. One method is the so-called separated MLE (SMLE). This maximizes the conditional likelihood of the experimental data given the simulation data. In this case, the GP parameters $\underline{\theta}$, $\underline{\beta}$, and σ^2 are determined by the marginal MLE from the computer data. These estimates are then inserted into in obtaining the conditional MLE. This result is subsequently maximized with respect to $\boldsymbol{\tau}$ and γ_E, to acquire estimates of the above parameters. Advantages of likelihood-based methods for calibration are that they simultaneously or conditionally utilize the combined data, and so enrich the calibration methods. The SMLE was demonstrated to be superior to the full MLE in Reference [1]. Thus, in this study, we compare the SMLE with the proposed method. Further details of the SMLE are presented in Appendix C.

4. Proposed Method: EM-Based Calibration

An EM algorithm is an iterative method for determining the MLE or maximum a posteriori (MAP) estimates of parameters in statistical models [32], where the model usually depends on the unobserved latent variables. One example of the unobserved latent variables is the calibration parameters in computer experiments. The EM iteration alternates between an expectation (E) step and a maximization (M) step. The E-step calculates the expected value of the log likelihood at the current parameter estimates provided from the M-step. The M-step computes the parameters by maximizing the expected log likelihood determined in the E-step. These parameter estimates are subsequently employed to determine the distribution of the latent variables (or calibration parameters) in the following E-step [32].

The EM algorithm was described and given its name in 1977 in a paper by Dempster et al. [33]. Since then, the EM algorithm has been applied to many research areas, including computational statistics, machine learning, computer vision, hidden Markov models, item response theory, and computed tomography. See References [34–36] for further information on the EM algorithm and its applications.

To date, the EM algorithm has not been applied to the calibration problem of complex computer code. Because the tuning parameters in real experiments can be treated as unobserved latent variables, an EM algorithm may be appropriate to obtain the distribution of the tuning parameters. The steps of the proposed method for a given GP model are presented as follows:

Initialization: Provide initial values ($\widehat{\boldsymbol{\tau}}$) for $\boldsymbol{\tau}$ from prior information on $\boldsymbol{\tau}$.
M-step: Determine the MLE ($\hat{\psi}$) of the GP parameters from the combined data in which $\widehat{\boldsymbol{\tau}}$ are inserted into the experimental data.
E-step: Set $\widehat{\boldsymbol{\tau}}$ as the conditional expectation of $\boldsymbol{\tau}$ given the estimates ($\hat{\psi}$ from the M-step) of the GP parameters obtained under the combined data.
Iterate: E- and M-steps until convergence.

In the k-th iteration of the E-step, the conditional expectation of $\boldsymbol{\tau}$ is actually the expectation of the posterior of $\boldsymbol{\tau}$:

$$\begin{aligned} E(\boldsymbol{\tau}^{*(\boldsymbol{k})}|\boldsymbol{y};\boldsymbol{x},\Psi^{(k)}) &= \int \boldsymbol{\tau}\ f(\boldsymbol{\tau}|\boldsymbol{y};\boldsymbol{x},\Psi^{(k)})\ d\boldsymbol{\tau} \\ &= \int \boldsymbol{\tau}\ \frac{f(\boldsymbol{\tau},\boldsymbol{y};\boldsymbol{x},\Psi^{(k)})}{f(\boldsymbol{y};\boldsymbol{x},\Psi^{(k)})}\ d\boldsymbol{\tau} \\ &= \int \boldsymbol{\tau}\ \frac{f(\boldsymbol{y}|\boldsymbol{\tau};\boldsymbol{x},\Psi^{(k)})p(\boldsymbol{\tau};\boldsymbol{x},\Psi^{(k)})}{\int f(\boldsymbol{\tau};\boldsymbol{x},\Psi^{(k)})p(\boldsymbol{\tau};\boldsymbol{x},\Psi^{(k)})\ d\boldsymbol{\tau}} d\boldsymbol{\tau} \\ &= \int \boldsymbol{\tau}\ \frac{G\times p(\boldsymbol{\tau};\boldsymbol{x},\Psi^{(k)})}{\int G\times p(\boldsymbol{\tau};\boldsymbol{x},\Psi^{(k)})d\boldsymbol{\tau}}\ d\boldsymbol{\tau} \\ &\Longrightarrow \boldsymbol{\tau}^{*(\boldsymbol{k+1})}, \end{aligned} \tag{11}$$

where $G = f(\underline{y}_B|\boldsymbol{\tau};\underline{x}_B,\hat{\psi}^{(k)})$ = pdf (probability density function) of $MN(F_B\widehat{\beta}^{(k)},\ \hat{\sigma}^{2(k)}V_B{}^{(k)})$, $\psi^{(k)} = (\underline{\theta}^{(k)},\underline{\beta}^{(k)},\sigma^{2(k)}\gamma_E^{2(k)})$, and $p(\boldsymbol{\tau};\underline{x}_B,\hat{\psi}^{(k)})$ is a prior (pdf) of $\boldsymbol{\tau}$. Note that F_B and V_B are functions of $\boldsymbol{\tau}$. A numerical integration method by Reference [37] in the R package “cubature” [38] was used for the calculation of (11). We set a uniform distribution as the prior of $\boldsymbol{\tau}$. Other priors can easily be incorporated.

A slightly modified likelihood function from (8) is employed to calculate the MLE in the M-step: $-2\ log\ L(\psi_B;\ \underline{y}_B, X_B, \widehat{\boldsymbol{\tau}})$, where $\widehat{\boldsymbol{\tau}}$ is obtained from the E-step. The M-step is basically the same as Step 3 in the max-min algorithm. The major difference between the two algorithms is that the max-min minimizes the ARSS, whereas the EM calculates the conditional expectation. One defect of the EM method is that it requires more computing time than the ALS or Kennedy–O'Hagan (KOH) method [2].

In each iteration of the E- and M-steps, parameters in the emulator and the combined data are updated. We expect this updating procedure affects positively to the estimation of the calibration parameter. The EM and max-min use the combined data for constructing an emulator, whereas the ALS uses the computer data only. The addition of relevant data generally enhances the prediction capacity of the emulator.

It is worth noting that the EM algorithm converges effectively, according to our experience when executing it in the test function study. The median iteration number is approximately 10 (the first and third quartiles are 6 and 28 iterations, respectively) based on 10 trials for test function 1, as described in detail in the following section.

Table 1 presents a classification of the calibration methods based on the $\boldsymbol{\tau}$ estimation, emulator building, and outer iteration. It can be observed that the EM algorithm can be viewed as an extension of the KOH method.

Table 1. A classification of the calibration methods based on $\boldsymbol{\tau}$ estimation, emulator building, and the number of outer iterations. Acronyms: NLSE = nonlinear least squares estimation, ALS = approximated NLSE, MLE (A) = maximum likelihood estimation calculated from data A, FMLE = full MLE, SMLE = separated MLE, Cond Expect (D) = conditional expectation based on data D, EM = expectation and maximization algorithm, Max-min = max-min algorithm described in Section 3.3, KOH = Kenney-O'Hagan method.

Method	$\boldsymbol{\tau}$ Estimation	Emulator Building	Outer Iterations
NLSE	NLSE		1
ALS	ALS (E)	MLE (C)	1
FMLE	MLE (C, E, ψ_B) (all together)		1
SMLE	MLE (C|E)	MLE (C)	1
EM	Cond Expect (C, E)	MLE (C, E)	many
Max-min	ALS (E)	MLE (C, E)	many
KOH	Cond Expect (E)	MLE(C)	1

5. Test Function Study

In this section, we describe the application of the calibration methods to test functions (or toy models) in which the true tuning parameters were known a priori. A set of five toy models in different situations were arranged for a comparison of the methods. These test functions are simple toy models, i.e., easy to compute. However, we treated these functions as if they were time-consuming simulators.

The computer data and experimental data with sample sizes n_C and n_E, respectively, were generated by

$$y_C = Y(\boldsymbol{\tau}_C, \mathbf{x}_C), \quad y_E = Y(\boldsymbol{\tau}^*, \mathbf{x}_E) + e.$$

The five test functions along with $n_E = n_C = 30$ and with the true constants of $\boldsymbol{\tau}$ are described as follows:

$$\begin{aligned}\textbf{Test function 1}: \ & Y(\tau, x) = \tau_1 \exp(\tau_2 + x_1) + \tau_1 x_2^2 - \tau_2 x_3^2 \\ \text{Computer data}: \ & T_1 \sim U(0,5),\ T_2 \sim U(0,4), \\ & x_1 \sim U(-3,3),\ x_2 \sim U(-3,3),\ x_3 \sim U(0,6) \\ \text{Experimental data}: \ & \tau_1 = 2,\ \tau_2 = 2,\ \sigma_E^2 = 1.\end{aligned}$$

$$\begin{aligned}
\textbf{Test function 2}: \;& Y(\tau, x) = \tau_1 \exp(\tau_2 + x_1 + \tau_3) + \tau_1\tau_3 x_2^2 - \tau_2 x_3^2 - \tau_3 \log(x_4) \\
\text{Computer data}: \;& T_1 \sim U(0,5),\; T_2 \sim U(0,4),\; T_3 \sim U(1,5), \\
& x_1 \sim U(-3,4),\; x_2 \sim U(-3,3),\; x_3 \sim U(0,6),\; x_4 \sim U(1,5) \\
\text{Experimental data}: \;& \tau_1 = 2,\; \tau_2 = 1,\; \tau_3 = 3,\; \sigma_E^2 = 1.
\end{aligned}$$

$$\begin{aligned}
\textbf{Test function 3}: \;& Y(\tau, x) = \tau_1 \exp(|x_1 + x_2|) + \tau_2(x_4 + 1.2x_5 + 1)/2.5 + \tau_3(\tau_2 + 2x_3 + x_4) \\
& + \tau_4(x_1 + x_3 - \tau_4 x_5 - x_6) + 2\cos(6(x_2 + x_3 + x_3)) \\
\text{Computer data}: \;& T_1 \sim U(0,5),\; T_2 \sim U(0,3),\; T_3 \sim U(0,7),\; T_4 \sim U(0,6), \\
& x_1 \sim U(-2,6),\; x_2 \sim U(-6,6),\; x_3 \sim U(-5,6),\; x_4 \sim U(-6,6) \\
& x_5 \sim U(-6,4),\; x_6 \sim U(-6,3) \\
\text{Experimental data}: \;& \tau_1 = 2,\; \tau_2 = 1,\; \tau_3 = 4,\; \tau_4 = 3,\; \sigma_E^2 = 6.
\end{aligned}$$

$$\begin{aligned}
\textbf{Test function 4}: \;& Y(\tau, x) = \tau_1 x_1^2 + \tau_2 x_2 + \tau_3 cos(x_3\pi) + \tau_4 sin(x_4\pi) \\
\text{Computer data}: \;& T_1 \sim U(0,5),\; T_2 \sim U(0,5),\; T_3 \sim U(0,7),\; T_4 \sim U(0,5) \\
& x_1 \sim U(0,3),\; x_2 \sim U(0,3),\; x_3 \sim U(0,2),\; x_4 \sim U(0,2) \\
\text{Experimental data}: \;& \tau_1 = 1,\; \tau_2 = 2,\; \tau_3 = 3,\; \tau_4 = 2,\; \sigma_E^2 = 4.
\end{aligned}$$

$$\begin{aligned}
\textbf{Test function 5}: \;& Y(\tau, x) = exp(\tau_2 x_1 + x_2)/(3 + \tau_1) + \tau_2\sqrt{(x_4 + x_5)} + \tau_1(x_3 + x_6)^2 \\
\text{Computer data}: \;& T_1 \sim U(0,5),\; T_2 \sim U(1,5) \\
& x_1 \sim U(-0.5,0.5),\; x_2 \sim U(-0.5,0.5),\; x_3 \sim U(-0.5,0.5), \\
& x_4 \sim U(0,2)\; x_5 \sim U(0,2),\; x_6 \sim U(0,1) \\
\text{Experimental data}: \;& \tau_1 = 2,\; \tau_2 = 2,\; \sigma_E^2 = 0.25.
\end{aligned}$$

Optimal Latin hypercube designs [39,40] were used for sampling in the independent variables for real experiments ($\mathbf{x}_E$) and for computer experiments ($\boldsymbol{\tau}_S, \mathbf{x}_S$). A total of 30 different designs for computer data were employed for each test function to take into account uncertainty in the design, whereas the real experimental design was fixed.

As a result, the average of estimates and the standard deviations from 30 trials are presented. The averaged Euclidean distance from the estimates to the true values was computed to evaluate the performance of the methods. (In addition to the Euclidean distance, one can consider an weighted distance such as the Mahalanobis distance [41]. It may be more meaningful than the Euclidean distance in the sense that the Mahalanobis distance takes into account the covariances among estimates of calibration parameters. The weighted distance is not considered in this study, but may be useful in the future study.) The root mean squared error (RMSE) of the estimates is also provided. The formula for the RMSE is as follows:

$$RMSE(\hat{\boldsymbol{\tau}}) = \sqrt{Bias^2(\hat{\boldsymbol{\tau}}) + \sum_{i=1}^{q}(std(\hat{\tau}_i))^2}, \tag{12}$$

where $Bias(\hat{\boldsymbol{\tau}})$ is the averaged Euclidean distance to the true constants and $std(\hat{\tau}_i)$ is the standard deviation of each estimate obtained from 30 replications.

Tables 2–6 present the results for each test function. The averaged Euclidean distance from the estimates to the true constants and the RMSEs of the estimates are displayed in each table. The standard deviations are the numbers in parentheses. In terms of the RMSE, the proposed method offered superior results over the ALS, SMLE, and max-min methods. In particular, the EM shows less bias and lower RMSE than the other methods. We have this result again in Figures 1–5, which present the boxplots of the distance to the true value for each of the five test functions. In many cases, the medians of the EM estimates were nearer to the true constants than those from the SMLE, ALS, and max-min methods. The box lengths of the EM estimates were shorter than those obtained from the other methods. It is notable that the max-min was more effective than the ALS and SMLE in test functions 2–4. One plausible reason that the EM was superior to the max-min is that the numerical integration for the conditional expectation in the EM may be more stable than the numerical optimization of the ARSS in the max-min.

The computing times of the max-min and EM algorithms were much longer than those of the ALS and SMLE (see Table 7). The times were obtained using a personal computer with an Intel i5 CPU (3.6 GHz) and 16 Giga bytes of memory. Thus, we faced the limitation in extending our test functions to a higher-dimensional calibration parameters because of the heavy computing time in the max-min and EM algorithms. Numerical integration method for more than 10 dimensions may not be practical in the EM algorithm. In such high-dimensional cases, a Monte Carlo integration technique would be useful, but it requires more computing time.

Table 2. Result for the test function 1 with true constants $\tau_1^* = 2$, $\tau_2^* = 2$. The values presented are the averaged estimates of τ^*, the averaged Euclidean distance from the estimates to the true constants, the root mean squared error (RMSE) of the estimates. The standard deviations are in parentheses. The acronyms are same as Table 1. The number 1 or 2 following each method stands for the Gaussian process Model 1 or Model 2.

Method	Average of $\widehat{\tau}_1$	Average of $\widehat{\tau}_2$	Average Distance	RMSE
ALS 1	1.609 (0.207)	1.773 (0.246)	0.527 (0.168)	0.617
ALS 2	1.448 (0.249)	1.602 (0.351)	0.749 (0.288)	0.864
SMLE 1	1.790 (0.245)	1.758 (0.275)	0.462 (0.147)	0.591
SMLE 2	2.228 (0.409)	1.465 (0.267)	0.703 (0.280)	0.856
Max-min 1	2.296 (0.135)	1.966 (0.224)	0.377 (0.109)	0.459
Max-min 2	2.382 (0.399)	2.151 (0.575)	0.718 (0.342)	1.003
EM 1	2.024 (0.212)	1.828 (0.204)	**0.276** (0.197)	**0.103**
EM 2	2.098 (0.234)	2.894 (0.166)	**0.264** (0.177)	**0.390**

Table 3. Same as Table 2 but for test function 2 with true values $\tau_1^* = 2$, $\tau_2^* = 1$, $\tau_3^* = 3$.

Method	Average of $\widehat{\tau}_1$	Average of $\widehat{\tau}_2$	Average of $\widehat{\tau}_3$	Average Distance	RMSE
ALS 1	1.801 (0.324)	1.349 (0.225)	3.301 (0.319)	0.598 (0.232)	0.784
ALS 2	1.715 (0.244)	1.107 (0.322)	3.034 (0.344)	0.570 (0.209)	0.779
SMLE 1	1.737 (0.565)	1.080 (0.527)	2.957 (0.519)	0.873 (0.398)	1.276
SMLE 2	1.912 (0.347)	0.901 (0.381)	2.853 (0.416)	0.585 (0.353)	0.884
Max-min 1	1.706 (0.293)	0.842 (0.127)	3.003 (0.435)	0.535 (0.327)	0.760
Max-min 2	1.768 (0.231)	1.106 (0.314)	3.054 (0.283)	0.511 (0.176)	0.702
EM 1	2.108 (0.238)	1.207 (0.116)	2.975 (1.164)	**0.371** (0.109)	**0.484**
EM 2	2.163 (0.239)	1.261 (0.160)	2.980 (0.112)	**0.414** (0.128)	**0.516**

Table 4. Same as Table 2 but for test function 3 with true values $\tau_1^* = 2$, $\tau_2^* = 1$, $\tau_3^* = 4$, $\tau_4^* = 3$.

Method	Average of $\widehat{\tau}_1$	Average of $\widehat{\tau}_2$	Average of $\widehat{\tau}_3$	Average of $\widehat{\tau}_4$	Average Distance	RMSE
ALS 1	2.314 (0.408)	1.185 (0.335)	3.911 (0.578)	3.367 (0.571)	0.963 (0.514)	1.366
ALS 2	2.224 (0.587)	1.175 (0.341)	3.713 (0.865)	3.323 (0.793)	1.324 (0.547)	1.895
SMLE 1	1.923 (0.465)	1.316 (0.472)	3.713 (0.515)	3.309 (0.648)	1.080 (0.443)	1.513
SMLE 2	1.861 (0.446)	1.365 (0.643)	3.423 (0.749)	3.541 (0.655)	1.609 (0.347)	2.047
Max-min 1	2.060 (0.321)	1.166 (0.300)	3.626 (0.553)	3.315 (0.528)	**0.902**(0.463)	1.261
Max-min 2	2.039 (0.544)	1.469 (0.339)	3.753 (0.785)	2.845 (0.726)	1.208 (0.612)	1.736
EM 1	2.583 (0.198)	1.522 (0.119)	3.668 (0.250)	3.029 (0.254)	0.938 (0.139)	**1.030**
EM 2	2.480 (0.239)	1.403 (0.252)	3.681 (0.325)	3.024 (0.223)	**0.859** (0.158)	**1.007**

Table 5. Same as Table 2 but for test function 4 with true values $\tau_1^* = 1$, $\tau_2^* = 2$, $\tau_3^* = 3$, $\tau_4^* = 2$.

Method	**Average of $\hat{\tau}_1$**	**Average of $\hat{\tau}_2$**	**Average of $\hat{\tau}_3$**	**Average of $\hat{\tau}_4$**	**Average Distance**	**RMSE**
ALS 1	0.935 (0.237)	1.753 (0.396)	3.304 (0.908)	1.951 (0.610)	1.025 (0.628)	1.568
ALS 2	0.874 (0.250)	1.852 (0.414)	2.765 (0.846)	1.729 (0.672)	1.149 (0.456)	1.650
SMLE 1	0.845 (0.291)	1.975 (0.758)	2.875 (0.532)	1.852 (0.607)	1.069 (0.437)	1.566
SMLE 2	1.008 (0.515)	1.895 (0.895)	2.740 (0.475)	1.816 (0.953)	1.398 (0.53)	2.038
Max-min 1	0.562 (0.261)	2.155 (0.472)	3.087 (0.549)	2.049 (0.560)	0.955 (0.291)	1.348
Max-min 2	0.600 (0.230)	2.319 (0.549)	3.114 (0.749)	2.104 (0.407)	1.095 (0.362)	1.510
EM 1	1.160 (0.198)	2.017 (0.296)	3.383 (0.190)	2.418 (0.169)	**0.718** (0.134)	**0.841**
EM 2	0.921 (0.268)	1.863 (0.397)	3.379 (0.248)	2.498 (0.160)	**0.797** (0.149)	**0.976**

Table 6. Same as Table 2 but for test function 5 with true values $\tau_1^* = 2$, $\tau_2^* = 3$.

Method	**Average of $\hat{\tau}_1$**	**Average of $\hat{\tau}_2$**	**Average Distance**	**RMSE**
ALS 1	1.796 (0.274)	2.076 (0.130)	0.361 (0.078)	0.471
ALS 2	1.745 (0.253)	2.060 (0.105)	0.365 (0.093)	0.456
SMLE 1	2.025 (0.307)	2.038 (0.116)	0.287 (0.157)	0.436
SMLE 2	2.157 (0.483)	2.041 (0.154)	0.436 (0.295)	0.667
Max-min 1	2.074 (0.374)	2.001 (0.100)	0.340 (0.192)	0.515
Max-min 2	1.889 (0.292)	2.045 (0.125)	0.298 (0.152)	0.436
EM 1	2.054 (0.140)	2.085 (0.098)	**0.189** (0.054)	**0.255**
EM 2	2.030 (0.143)	2.102 (0.085)	**0.172** (0.094)	**0.239**

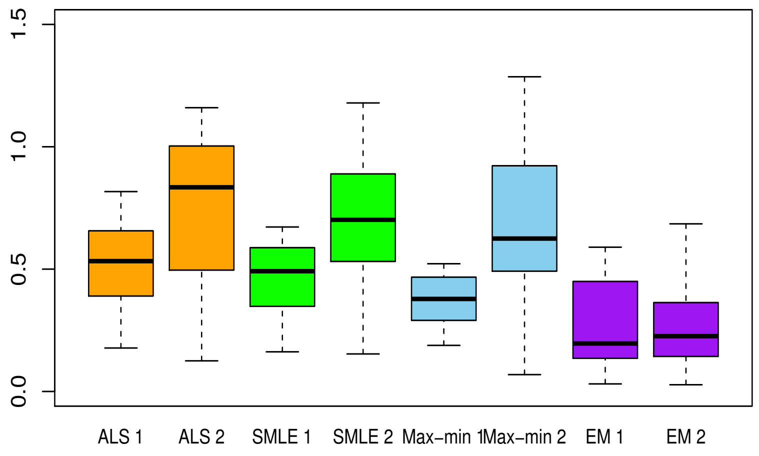

Figure 1. Boxplots of the Euclidean distances between true constants and estimated values from the calibration methods for test function 1, calculated from 30 random trials. The number 1 or 2 following each method stands for the Gaussian process Model 1 or Model 2. ALS: approximated nonlinear least squares estimation, SMLE: separated maximum likelihood estimation, EM: expectation and maximization algorithm, Max-min: max-min algorithm described in Section 3.3.

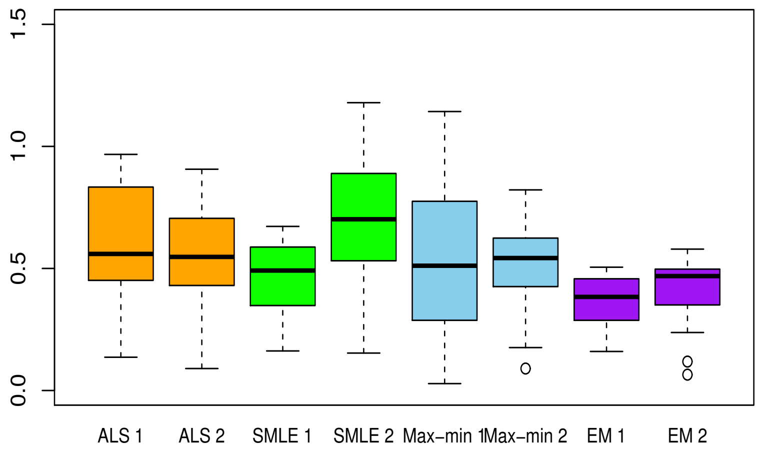

Figure 2. Same as Figure 1 but for test function 2.

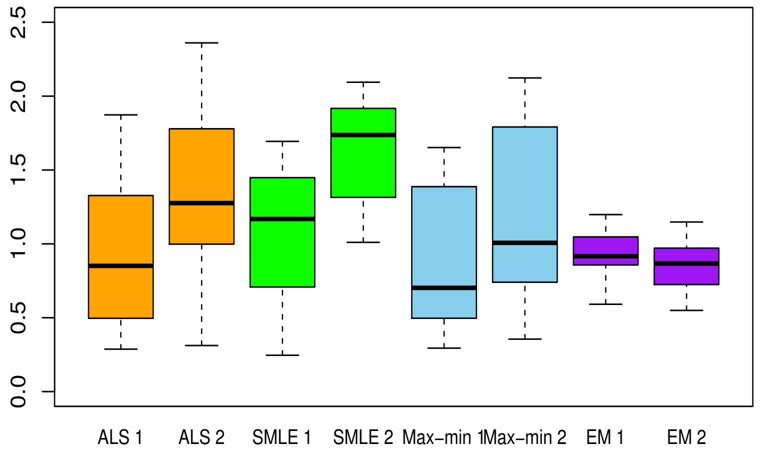

Figure 3. Same as Figure 1 but for test function 3.

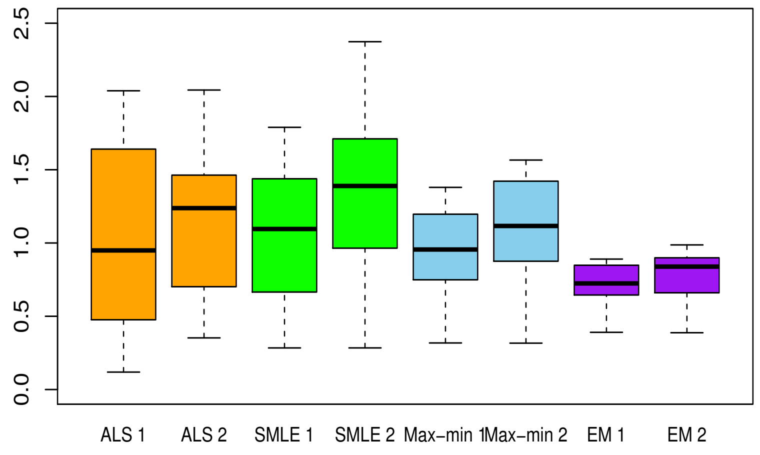

Figure 4. Same as Figure 1 but for test function 4.

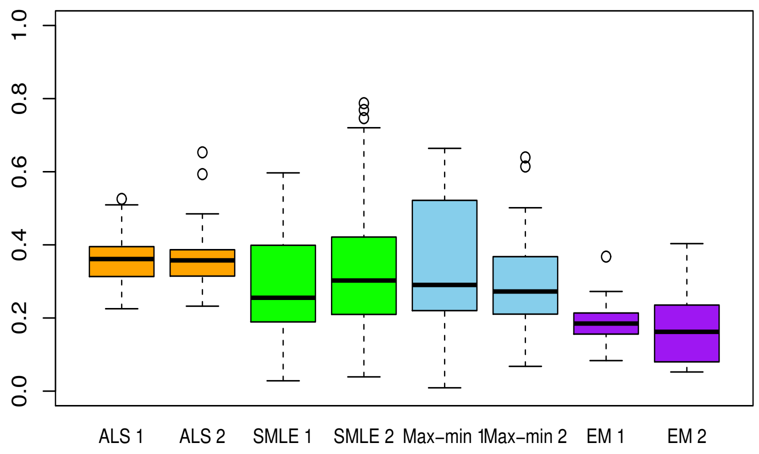

Figure 5. Same as Figure 1 but for test function 5.

Table 7. Averaged computing time in seconds for tuning methods with $n_E = n_C = 30$ from 30 trials for each of test functions 2, 4, and 5.

Function	**No. of τ**	**No. of X**	**ALS**	**SMLE**	**Max-Min**	**EM**
Test 2	3	4	149	150	661	471
Test 4	4	6	185	295	1207	1300
Test 5	2	6	23	117	435	320

6. Application to Nuclear Fusion Simulator

We present, in this section, the application of the calibration methods to computer code known as "Baldur" [42], which is a time-consuming simulator for the energy confinement time in a nuclear fusion device. It is called a "tokamak" in the Russian language. The mathematical model is simply expressed as follows, for a known complex function f calculated from the Baldur code:

$$y = f(\tau_1, \tau_2, \tau_3, \tau_4, x_1, x_2, x_3, x_4), \tag{13}$$

where x_1 is the input heating power, x_2 is the toroidal plasma current, x_3 is the line average electron density, and x_4 is the toroidal magnetic field. Calibration parameters $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3, \tau_4)$ determine the energy transfer, where each parameter is respectively, related to drift waves, rippling, resistive ballooning, and the critical value of η_i [42].

The experimental data consisted of $\mathbf{x} = \{x_1, x_2, x_3, x_4\}$ with sample size 42 from the Poloidal Divertor Experiment (PDX) tokamak at Princeton. The computer data consisted of $(\boldsymbol{\tau}, \mathbf{x})$ with sample size 64 from the Baldur code. The details on data can be found in References [1,42].

Table 8 presents the results of the $\boldsymbol{\tau}$ estimation when using the ALS, SMLE, max-min, and EM algorithms on the basis of GP Model 1 and Model 2 from the tokamak data. The results were obtained using R program on a personal computer with an Intel i5 CPU (3.6 GHz) and 16 Giga bytes of memory.

Table 8. Estimates of the tuning parameters by various calibration methods from the nuclear fusion data. The last column shows computing time in seconds for tuning methods. The acronyms are same as Figure 1.

Method	$\hat{\tau}_1$	$\hat{\tau}_2$	$\hat{\tau}_3$	$\hat{\tau}_4$	Computing Time
ALS 1	1.012	2.035	1.110	1.308	337
ALS 2	1.195	1.593	0.956	1.035	1017
SMLE 1	1.120	2.055	0.118	1.303	585
SMLE 2	1.290	1.342	1.204	1.157	1230
Max-min 1	0.670	1.125	0.468	1.826	1674
Max-min 2	0.732	2.592	1.311	1.034	8042
EM 1	−0.459	3.227	1.750	1.407	1481
EM 2	1.064	2.428	1.048	1.030	7717

Figure 6 depicts the residuals ($y_E - \hat{Y}(T_E)$) plotted according to the predicted values $\hat{Y}(T_E)$ that were obtained by various methods using GP Model 1 and Model 2. The residual plots from all methods exhibited linear trends. The trend for the EM algorithm was the lowest among the methods.

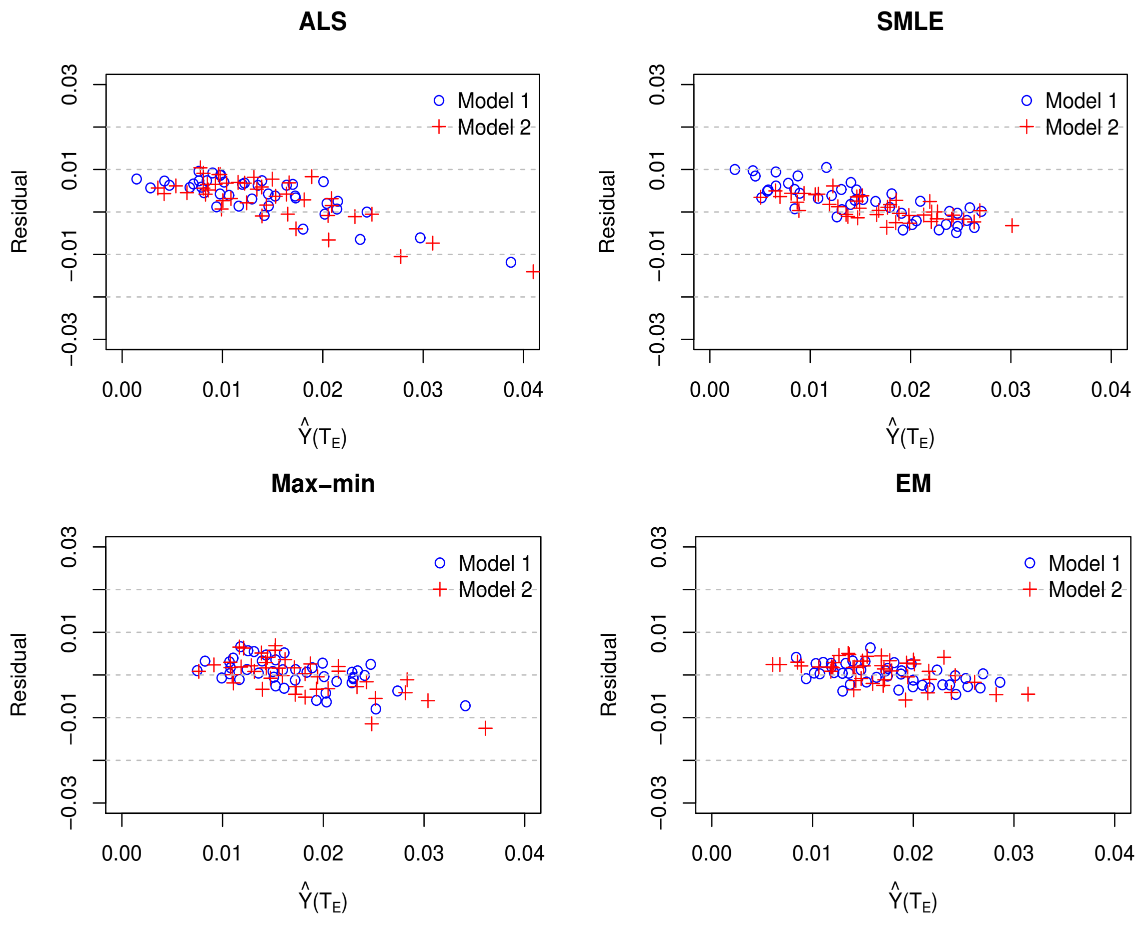

Figure 6. Residual plots for nuclear fusion data in which the simulation code was calibrated by various methods (ALS, SMLE, Max-min, and EM) using Gaussian process Model 1 or Model 2. The X-axes represents the predicted values $\hat{Y}(T_E)$ and the Y-axes represents the residuals ($y_E - \hat{Y}(T_E)$). The acronyms are same as Figure 1.

7. Discussion

Certain basic limitations exist when calibrating computer models to real data. We have experienced that the performance of the calibration methods was influenced significantly by the designs of both the physical and computer experiments [43,44]. Thus, a sequential designing approach must be very useful in practice [10,11,16]. Relevant experimental designs under a sequential tuning will improve the calibration very well.

Iterative versions of calibration methods, including the KOH method, may be available. For example, an iterative version of the SMLE method can be summarized as follows:

Step 1: Acquire $\widehat{\boldsymbol{\tau}}$ by maximizing the conditional likelihood function of $\boldsymbol{\tau}$ on the E-data, given $(y_C,\ \hat{\psi}_C)$.
Step 2: [maximization] Acquire $\hat{\psi}_B$ by maximizing the likelihood from the both data, in which $\widehat{\boldsymbol{\tau}}$ are inserted into the E-data.
Step 3: [maximization] Acquire $\widehat{\boldsymbol{\tau}}$ by maximizing the conditional likelihood function of $\boldsymbol{\tau}$ on the E-data, given $(y_C,\ \hat{\psi}_B)$.
Step 4: Iterate Steps 2 and 3 until convergence.

The estimates $\widehat{\tau}$ may vary according to the selected emulator. Thus an importance in calibration is the selection of the GP surrogate with some regression variables and GP parameters. We only used simple models (6) in this study. If the optimal GP model is built by the model selection algorithm, the result could be different.

8. Summary

The ALS method using a GP emulator is a basic calibration technique for complex computer models. However, it exhibits the potential drawback that the emulator is constructed once and it is no longer re-built. To overcome this defect, an iterative (EM) algorithm has been proposed in this study. The calibration parameters of the simulation code are calculated by the conditional expectation (E-step), whereas the GP parameters are updated by maximum likelihood estimation (M-step). These EM steps are alternately repeated until convergence by using both computer and experimental data.

We examined five test functions for a performance comparison. We confirmed that our proposed method (the EM algorithm) provided better results than the SMLE, ALS, and max-min methods in terms of the RMSE. The disadvantage of the proposed method is that it is more time-consuming than the ALS, because the EM algorithm needs to optimize complex functions and compute the conditional expectation using numerical integration based on the combined data. Nonetheless, the EM method can provide improved calibration as well as a superior emulator of the computer code compared to non-iterative methods, including the ALS and SMLE.

Author Contributions: Conceptualization, Y.A.S. and J.-S.P.; methodology, Y.A.S. and J.-S.P.; software, Y.A.S.; data curation, J.-S.P.; writing–original draft preparation, Y.A.S.; writing–review and editing, J.-S.P.; visualization, Y.A.S.; supervision, J.-S.P.; funding acquisition, Y.A.S. and J.-S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Korea Meteorological Administration Research and Development Program "Development of the AI technique for the prediction of rainfall optimized over the Korean peninsula" under Grant (KMA2018-00124). Park's work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT No.2016R1A2B4014518 and 2020R1I1A3069260)..

Acknowledgments: The authors would like to thank the reviewers and the guest editors of the special issue for helpful comments and suggestions, which have significantly improved the presentation of this paper. This report is a part of an invited talk by the second author at the GdR Mascot-Num annual conference at Ecole de Mines St-Etienne, France in 2015. He would like to thank the organizers of that conference for their hospitality. We are grateful to Professor Clifford Singer (Dept. Nuclear Engineering, Univ. of Illinois at Urbana-Champaign) for providing the Tokamak data.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Abbreviations

NLSE	non-linear least squares estimation
GP	Gaussian process
ALS	approximated NLSE
ARSS	approximated residual sum of squares
E-data	real experimental data
C-data	computer data
B-data	both of E-data and C-data
MLE	maximum likelihood estimation
FMLE	full MLE
SMLE	separated MLE
EM	expectation and maximization
E-step	expectation step
M-step	maximization step
pdf	probability density function
Max-min	maximization and minimization algorithm
KOH	Kenney-O'Hagan method
RMSE	root mean squared error

Appendix A. Maximum Likelihood Estimation in GP Model

Once the data have been obtained at the design sites $\{\underline{x}_1, ..., \underline{x}_n\}$, we employ the MLE method to estimate the GP parameters. Because we assume that $y(x)$ is a GP with mean $F\underline{\beta}$ and covariance $\sigma^2\, R$ for a design matrix F, the likelihood function of $\underline{y}$ is

$$L(\underline{y};\ \underline{\theta}, \underline{\beta}, \sigma^2, \gamma_C,\ \underline{x}) \ = \ \frac{(2\pi\sigma^2)^{-n/2}}{\sqrt{|\ V|}}\, exp\left(\ - \ \frac{(\underline{y} - F\underline{\beta})^t V^{-1} (\underline{y} - F\underline{\beta})}{2\, \sigma^2} \right). \tag{A1}$$

When the hyper-parameters $\underline{\theta}$ and γ_C are specified, the MLEs of σ^2 and $\underline{\beta}$ are

$$\widehat{\underline{\beta}} \ = \ (F^t V^{-1} F)^{-1} F^t V^{-1} \underline{y}, \quad \widehat{\sigma^2} \ = \ \frac{1}{n} (\underline{y} - F\widehat{\underline{\beta}})^t V^{-1} (\underline{y} - F\widehat{\underline{\beta}}). \tag{A2}$$

Because there is not a closed-form solution from the likelihood equations, we used a numerical optimization routine. The Cholesky decomposition $V = U^t U$ is used as a major computation in calculating the likelihood function, where U is an upper triangular Cholesky factor.

Appendix B. Data Structure and Notations for Calibration

Appendix B.1. Notations

The following notations are used for the computer and experimental data with n_C and n_E observations:

T: input variables $T = (\boldsymbol{\tau}, \mathbf{x})$ of computer code with d dimensions
$\boldsymbol{\tau}$: q-dimensional calibration parameters
$\boldsymbol{\tau}_C$: q-dimensional input variables of computer code corresponding to calibration parameters $\boldsymbol{\tau}$
$\mathbf{x}_C$: d-q dimensional independent variables in computer code
$Y(\boldsymbol{\tau}, \mathbf{x})$ or y_C: responses from simulator for input variables $(\boldsymbol{\tau}, \mathbf{x})$
$\mathbf{x}_E$: independent variables in real experiments with d-q dimensions
y_E: observations in real experiments

The subscripts C and E indicate the computer and real experiments, respectively. If the simulator mimics the real experimental data effectively well without discrepancy, we use the following model to approximate y_E:

$$y_E = Y(\boldsymbol{\tau}, \mathbf{x}_E) + e, \tag{A3}$$

where e is independent and normally distributed random variable with mean zero and variance σ_E^2. Certain contents of this Appendix are similar to those in References [4,6].

Appendix B.2. Computer and Experimental Data

We have the data matrix of the independent variables X_C and X_E for the computer and experimental data, respectively:

$$X_E = \begin{bmatrix} \tau_1 & \tau_2 & \cdots & \tau_q & x_{E11} & x_{E21} & \cdots & x_{Ep1} \\ \tau_1 & \tau_2 & \cdots & \tau_q & x_{E12} & x_{E22} & \cdots & x_{Ep2} \\ \vdots & & & \vdots & & & & \vdots \\ \tau_1 & \tau_2 & \cdots & \tau_q & x_{E1n_E} & x_{E2n_E} & \cdots & x_{Epn_E} \end{bmatrix} \tag{A4}$$

$$X_C = \begin{bmatrix} t_{11} & t_{21} & \cdots & t_{q1} & x_{C11} & x_{C21} & \cdots & x_{Cp1} \\ t_{12} & t_{22} & \cdots & t_{q2} & x_{C12} & x_{C22} & \cdots & x_{Cp2} \\ \vdots & & & \vdots & & & & \vdots \\ t_{1n_C} & t_{2n_C} & \cdots & t_{qn_C} & x_{C1n_C} & x_{C2n_C} & \cdots & x_{Cpn_C} \end{bmatrix}. \tag{A5}$$

In the above, t_{ij} in X_C represents the j-th value of the i-th T variable (T_i), whereas x_{Eij} and x_{Cij} denote the j-th value of the i-th X variable of the experimental (X_{Ei}) and computer (X_{Ci}) input. Furthermore, X_E is a $n_E \times (q+p)$ matrix and X_C is a $n_C \times (q+p)$ matrix. Note that the first part of X_E is composed of the unknown parameters $\tau_1, \cdots, \tau_q$, whereas the corresponding part of X_C comprises the input values (t_{ij}).

Appendix B.3. Combined Data

The following notations are introduced for the combined computer and experimental data:

$$X_B = \begin{pmatrix} X_C \\ X_E \end{pmatrix}, \quad F_B = \begin{pmatrix} F_C \\ F_E \end{pmatrix} = \begin{pmatrix} f(X_C) \\ f(X_E) \end{pmatrix}, \quad \underline{y}_B = \begin{pmatrix} \underline{y}_C \\ \underline{y}_E \end{pmatrix}. \tag{A6}$$

X_B is the design matrix of the independent variables. F_B is defined as the functions of the input variable values. $\underline{y}_B$ is a vector of the combined responses. Note that $\underline{T}$ are the input varaibles of the computer code. In this case, X_C and F_C contain $\underline{T}$, whereas X_E and F_E are the functions of the tuning parameters $\boldsymbol{\tau}$.

The GPM is subsequently applied to the computer and experimental data simultaneously. Let $\psi = (\boldsymbol{\tau}, \underline{\theta}, \gamma_C, \gamma_E, \sigma^2, \underline{\beta})$, where $\gamma_C = \sigma^2_{\epsilon C}/\sigma^2$ and $\gamma_E = \sigma^2_{\epsilon E}/\sigma^2$. Here, $\sigma^2_{\epsilon C}$ and $\sigma^2_{\epsilon E}$ are the variances of the error term (ϵ) in the GP model for the computer and experimental data, respectively. We set $\sigma^2_{\epsilon C} = 0$, and thus, $\gamma_C = 0$, because only a deterministic computer model is considered in this study. When necessary, $\underline{\beta}_C$ and $\underline{\beta}_E$ are used to denote the regression coefficients for the computer and real experimental data. Thus, given the independence and normality assumptions, we have

$$Law\ (\underline{y}_B|\psi) = N(F_B\underline{\beta}_B,\ V_B), \tag{A7}$$

where

$$\underline{\beta}_B = (\underline{\beta}_C,\ \underline{\beta}_E)^t, \tag{A8}$$

$$V_B = \begin{bmatrix} V_{CC} & V_{CE} \\ V_{EC} & V_{EE} \end{bmatrix} = \sigma^2 \begin{bmatrix} R(X_C, X_C) & R(X_C, X_E) \\ R(X_E, X_C) & R(X_E, X_E) \end{bmatrix} + \sigma^2 \begin{bmatrix} \gamma_C I & 0 \\ 0 & \gamma_E I \end{bmatrix}, \tag{A9}$$

in which $R(X_C, X_E)$ represents a $n_C \times n_E$ correlation matrix computed between X_C and X_E. V_B is a $n_B \times n_B$ positive definite covariance matrix for the combined data, where $n_B = n_C + n_E$.

Appendix C. Separated MLE for Calibration

For the details of the SMLE, we utilize the conditional distribution of the experimental data given the computer data, which is normally distributed with a mean

$$\mu_{E|C} \;=\; E[\underline{y}_E|\underline{y}_C;\, \boldsymbol{\tau},\psi] \;=\; F_E\underline{\beta}_E + + V_{CE}^t\, V_{CC}^{-1}\,(\underline{y}_C - F_C\underline{\beta}_C) \tag{A10}$$

and covariance

$$V_{E|C} \;=\; \mathrm{Cov}[\underline{y}_E|\underline{y}_C;\, \boldsymbol{\tau},\psi] \;=\; V_{EE} \,-\, V_{CE}^t\, V_{CC}^{-1}\, V_{CE}, \tag{A11}$$

where the covariance matrices V are given as per (A9). In these formulae, we suppress the parameter dependencies in $\mu_E = F_E\underline{\beta}_E$, $\mu_C = F_C\underline{\beta}_C$, V_{CE}, V_{CC}, and V_{EE}. Thus, the minus two times concentrated (without constants) log conditional likelihood function with $\widehat{\underline{\beta}}$ and $\hat{\sigma}^2_{E|C}$ plugged in is

$$-\,2\; log\; L(\boldsymbol{\tau},\gamma_E;X_E,\underline{y}_E|\;\underline{y}_C,\hat{\gamma}_C,\widehat{\underline{\beta}}_C,\hat{\underline{\theta}},X_C) \;=\; n_E\; log\; \hat{\sigma}^2_{E|C} \,+\, log\; |V_{E|C}|, \tag{A12}$$

where

$$\hat{\sigma}^2_{E|C} = (\underline{y}_E - \hat{\mu}_{E|C})^t\; V_{E|C}^{-1}\;(\underline{y}_E - \hat{\mu}_{E|C})/n_E, \tag{A13}$$

$$\hat{\mu}_{E|C} = F_E\widehat{\underline{\beta}}_E \,+\, V_{CE}^t\; V_{CC}^{-1}\;(\underline{y}_C - F_C\widehat{\underline{\beta}}_C). \tag{A14}$$

References

1. Cox, D.; Park, J.S.; Singer, C.E. A statistical method for tuning a computer code to a data base. *Comput. Stat. Data Anal.* **2001**, *37*, 77–92. [CrossRef]
2. Kennedy, M.C.; O'Hagan, A. Bayesian Calibration of Computer Models. *J. R. Stat. Ser. B* **2001**, *63*, 425–464. [CrossRef]
3. Han, G.; Santner, T.J.; Rawlinson, J.J. Simultaneous Determination of Tuning and Calibration Parameters for Computer Experiments. *Technometrics* **2009**, *51*, 464–474. [CrossRef] [PubMed]
4. Seo, Y.A.; Lee, Y.; Park, J.S. Iterative method for tuning complex simulation code. *Commun. Stat-Simul. Comput.* **2020**.10.1080/03610918.2020.17283 [CrossRef]
5. Trucano, T.G.; Swiler, L.P.; Igusa, T.; Oberkampf, W.L.; Pilch, M. Calibration, validation, and sensitivity analysis: What's what. *Reliab. Eng. Syst. Saf.* **2006**, *91*, 1331–1357. [CrossRef]
6. Lee, Y.; Park, J.-S. Generalized Nonlinear Least Squares Method for the Calibration of Complex Computer Code Using a Gaussian Process Surrogate. *Entropy* **2020**, *22*, 985. doi:10.3390/e22090985 [CrossRef]
7. Gramacy, R.; Bingham, D.; Hollowa, J.; Grosskopf, M. Calibrating a large computer experiment simulating radiative shock hydrodynamics. *Ann. Appl. Stat.* **2015**, *9*, 1141–1168. [CrossRef]
8. Wong, R.; Storlie, C.; Lee, T. A frequentist approach to computer model calibration. *J. R. Stat. Soc. Ser. B* **2017**, *79*, 635–648. [CrossRef]
9. Goh, J.; Bingham, D.; Holloway, J.P.; Grosskopf, M.J.; Kuranz, C.C.; Rutter, E. Prediction and Computer Model Calibration Using Outputs From Multifidelity Simulators. *Technometrics* **2013**, *55*, 501–512. [CrossRef]
10. Damblin, G.; Barbillon, P.; Keller, M.; Pasanisi, A.; Parent, E. Adaptive numerical designs for the calibration of computer codes. *SIAM/ASA J. Uncertain. Quantif.* **2018**, *6*, 151–179. [CrossRef]
11. Kumar, A. Sequential tuning of complex computer models. *J. Stat. Comput. Simul.* **2015**, *85*, 393–404. [CrossRef]
12. Higdon, D.; Gattiker, J.; Williams, B.; Rightley, M. Computer Model Calibration Using High Dimensional Output. *J. Am. Stat. Assoc.* **2008**, *103*, 570–583. [CrossRef]
13. Tuo, R.; Wu, C.F.J. Prediction based om the Kennedy-O'Hagan calibration model: Asymptotic consistency and other properties. *Stat. Sin.* **2018**, *28*, 743–759.
14. Plumlee, M. Bayesian calibration of inexact computer models. *J. Am. Stat. Assoc.* **2017**, *112*, 1274–1285. [CrossRef]
15. Gu, M.; Wang, L. Scaled Gaussian stochastic process for computer model calibration and prediction. *SIAM/ASA J. Uncertain. Quantif.* **2018**, *6*, 1555–1583. [CrossRef]
16. Pratola, M.T.; Sain, S.R.; Bingham, D.; Wiltberger, M.; Riglerd, E.J. Fast sequential computer model calibration of large nonstationary spatial-temporal processes. *Technometrics* **2013**, *55*, 232–242. [CrossRef]
17. Pandit, R.; Infield, D. Gaussian Process Operational Curves for Wind Turbine Condition Monitoring. *Energies* **2018**, *11*, 1631. [CrossRef]
18. Albert, C.G. Gaussian Processes for Data Fulfilling Linear Differential Equations. *Proceedings* **2019**, *33*, 5. [CrossRef]
19. Molada-Tebar, A.; Riutort-Mayol, G.; Marqués-Mateu, Á.; Lerma, J.L. A Gaussian Process Model for Color Camera Characterization: Assessment in Outdoor Levantine Rock Art Scenes. *Sensors* **2019**, *19*, 4610. [CrossRef]

20. Kim, S.; Jung, S.; Baek, S.-M. A Model for Predicting Energy Usage Pattern Types with Energy Consumption Information According to the Behaviors of Single-Person Households in South Korea. *Sustainability* **2019**, *11*, 245. [CrossRef]
21. Oladyshkin, S.; Mohammadi, F.; Kroeker, I.; Nowak, W. Bayesian^3 Active Learning for the Gaussian Process Emulator Using Information Theory. *Entropy* **2020**, *22*, 890. [CrossRef] [PubMed]
22. Sacks, J.; Welch, W.; Mitchell, T.; Wynn, H. Design and analysis of computer experiment (with discussion). *Stat. Sci.* **1989**, *4*, 409–435. [CrossRef]
23. Liao, Y.; Xie, J.; Wang, Z.; Shen, X. Multisensor Estimation Fusion with Gaussian Process for Nonlinear Dynamic Systems. *Entropy* **2019**, *21*, 1126. [CrossRef]
24. Mahmoodian, M.; Torres-Matallana, J.A.; Leopold, U.; Schutz, G.; Clemens, F.H.L.R. A Data-Driven Surrogate Modelling Approach for Acceleration of Short-Term Simulations of a Dynamic Urban Drainage Simulator. *Water* **2018**, *10*, 1849. [CrossRef]
25. Mohammadi, H.; Challenor, P.; Goodfellow, M. Emulating dynamic non-linear simulators using Gaussian processes. *Comput. Stat. Data Anal.* **2019**, *139*, 178–196. [CrossRef]
26. Santner, T.J.; Williams, B.; Notz, W. *The Design and Analysis of Computer Experiments*, 2nd ed.; Springer: New York, NY, USA, 2018.
27. Roustant, O.; Ginsbourger, D.; Deville, Y. DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *J. Stat. Softw.* **2012**, *51*, 54. [CrossRef]
28. Welch, W.J.; Buck, R.J.; Sacks, J.; Wynn, H.P.; Mitchell, T.J.; Morris, M.D. Screening, predicting, and computer experiments. *Technometrics* **1992**, *34*, 15–25. [CrossRef]
29. Marrel, A.; Iooss, B.; Drope, F.V.; Volkova, E. An efficient methodology for modeling complex computer codes with Gaussian processes. *Comput. Stat. Data Anal.* **2008**, *52*, 4731–4744. [CrossRef]
30. Lee, Y.; Park, J.S. Model selection algorithm in Gaussian process regression for computer experiments. *Commun. Stat. Appl. Methods* **2017**, *24*, 383–396 [CrossRef]
31. Gramacy, R.B. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*; CRC Press: Boca Raton, FL, USA, 2020.
32. Wikipedia. Expectation–Maximization Algorithm. Available online: https://en.wikipedia.org/wiki/Expectation%E2%80%93 maximization_algorithm (accessed on 16 October 2020).
33. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Stat. Ser. B.* **1977**, *39*, 1–38.
34. Bilmes, J. *A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*; Technical Report ICSI-TR-97-021; University of Berkeley: Berkeley, CA, USA, 1997.
35. Gupta, M.R.; Chen, Y. Theory and Use of the EM Algorithm. *Found. Trends Signal Process.* **2010**, *4*, 223–296. [CrossRef]
36. McLachlan, G.J.; Krishnan, T. *The EM Algorithm and Extensions*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2008.
37. Genz, A.; Malik, A. Remarks on algorithm 006: An Adaptive Algorithm for Numeric Integration over an N-Dimensional Rectangular Region. *J. Comp. Appl. Math* **1980**, *6*, 295–302 [CrossRef]
38. Balasubramanian, N.; Manuel, K.; Steven, G.; Thomas, H.; Annie, B.; Kiên, K.; Simen, G. Cubature: Adaptive Multivariate Integration over Hypercubes 2.0.4.1. Available online: Https://CRAN.R-project.org/package=cubature (accessed on 13 October 2020).
39. Morris, M.; Mitchell, T. Exploratory designs for computational experiments. *J. Stat. Plan. Infer.* **1995**, *43*, 381–402. [CrossRef]
40. Carnell, R. lhs: Latin Hypercube Samples. R Package Version 0.16. Available online: Https://CRAN.R-project.org/package=lhs (accessed on 13 October 2020).
41. Johnson R.A.; Wichern D.W. *Applied Multivariate Statistical Analysis*, 6th ed.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2007.
42. Singer, C.E.; Post, D.E.; Mikkelsen, D.R.; Redi, M.H.; McKenney, A.; Silverman, A.; Seidl, F.G.P.; Rutherford, P.H.; Hawryluk, R.J.; Langer, W.D.; et al. BALDUR: A one-dimensional plasma transport code. *Comput. Phys. Commun.* **1988**, *49*, 275–398. [CrossRef]
43. Beck, J.; Guillas, S. Sequential design with mutual information for computer experiments (MICE): Emulation of a tsunami model. *J. Uncertain. Quantif.* **2016**, *4*, 739–766. [CrossRef]
44. Cailliez, F.; Bourasseau, A.; Pernot, P. Calibration of Forcefields for Molecular Simulation: Sequential Design of Computer Experiments for Building Cost-Efficient Kriging Metamodels. *J. Comput. Chem.* **2014**, *35*, 130–149. [CrossRef]