QBS
Vol. 39, No. 2, pp. 137-145 (2020)
https://doi.org/10.22283/qbs.2020.39.2.137
pISSN 2288-1344 eISSN 2508-7185 **Original Article**

Quantitative Bio-Science
Official Journal of the Institute of
Natural Science at Keimyung University
Available Online at https://qbs.kmu.ac.kr:442/

Estimation of the Universal Constants in Complex Computer Code Using Multiple Gaussian Process Models

Seung-Gu Kim[1], Tae Yoon Kim[2], Jeong-Soo Park[3,*]
[1]Department of Data and Information, Sangji University, Wonju 26339, Korea
[2]Department of Statistics, Keimyung University, Daegu 42601, Korea
[3]Department of Statistics, Chonnam National University, Gwangju 61186, Korea
(Received October 25, 2020; Revised November 10, 2020; Accepted November 15, 2020)

ABSTRACT

The approximated nonlinear least squares (ALS) method has been used for adjusting unknown universal constants in the complex simulation code, which is very time-consuming to execute. The ALS tunes or calibrates the computer code by minimizing the squared difference between real observations and computer output using a surrogate such as a Gaussian process model. A potential drawback of the ALS is that it does not take the uncertainty in the approximation of the simulation by a surrogate model into account. The calibration result is too dependent on selection of a surrogate model. To address these problems, we consider a simple ensemble method that averages the respective estimates from four different models. A total of three test functions in different conditions are examined for a comparative analysis. Based on the test function study, we find that the ensemble method by four models provides better results than the ALS method. We also review some calibration methods including a generalized ALS, an iterative version of ALS, and likelihood-based method. We provide a brief discussion for comparison and future direction.

Key words : Best linear unbiased predictor, Code tuning, Computer experiments, Generalized least squares estimator, Kriging, Numerical optimization

1. Introduction

In many areas of study, modern scientific researchers try to develop and use computer simulator instead of the physical experiments which are sometimes too expensive or impossible. However, there are usually unknown parameters similar to the gravitational constant. One of the classic methods of adjusting unknown universal constants in computer code is the nonlinear least squares estimation (NLSE), which minimizes the sum of the squared differences between the responses of computer simulations and the experimental observations. However, if the computer code is time-consuming to run, the NLSE becomes too expensive in terms of time. In these cases, one can use a statistical model to tune the parameters in the computer simulator such that the model can explain the physical experimental data very well. This procedure is called “code tuning” or “calibration” [1-4].

We define the calibration as the process of improving the agreement of a set of code calculations with respect to a chosen and fixed set of experimental data via adjustment of the parameters implemented in the simulator [5]. Han et al. [3] differentiated between tuning parameter and calibration parameter. In this study, however, the two parameters are treated as the same. We will be lazy by using those two words (calibration and code tuning) interchangeably [6].

---

* Correspondence should be addressed to Jeong-Soo Park, Professor, Department of Statistics, Chonnam National University, Gwanju 61186, Korea. Tel: +82-62-530-3445, Fax: +82-62-530-3449, E-mail: jspark@jnu.ac.kr
The first two authors should be regarded as joint first authors. Seung-Gu Kim is a professor, Department of Data and Information, Sangji University, Wonju 26339, Korea. Tae Yoon Kim is a professor, Department of Statistics, Keimyung University, Daegu 42601, Korea.

Cox *et al.* [1] studied an approximated NLSE (ALS) for code-tuning, in which they employed the Gaussian process (GP) as a metamodel of complex computer code. That is, the ALS first fits the GP model to the computer data and then it treats the fitted GP model as if it were true simulation code, which makes it computationally feasible. The GP has been successfully used to analyze computer experiments [7-9]. We adopt the GP model as a metamodel of computer simulation code. In this report, a metamodel or an emulator or a surrogate or a GP model are used as a same meaning.

Our work focuses on calibration in frequentist fashion [10, 11], rather than a Bayesian one [2,12,13]. Kennedy and O'Hagan [2] introduced a Bayesian calibration in which a bias correction is done by the GP. Recent contribution to this topic includes efficient designs for calibration [13], computer experiments for big data [10], multi-fidelity [12], sequential tuning [14], identifiability [15,16], and surrogate modelling [11,17,18].

This paper is organized as follows. Section 2 describes a GP for computer experiments. The ALS approach based on four models and its ensemble are introduced in Section 3. In Section 4, a test function study is presented. An iterative version of ALS method, a generalized ALS method, and likelihood-based approach are reviewed in Section 5. Summary and discussion are given in Section 6. Some contents of this study are inevitably similar to those of References [1,4,6] because the problem setting of these studies are same to this report.

2. Gaussian Process Model for Computer Experiments

In this section, we describe the statistical model for the computer simulation data based on the Gaussian process. In many cases, computer simulation code is deterministic. For this reason, Sacks *et al.* [7] proposed adopting the Gaussian process model (GPM) for computer experiments. The expression of the GP is as follows:

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

where $\{f_i(\cdot)\}$ is a set of known functions, β is a vector of unknown regression coefficients, and $Z(\cdot)$ is the Gaussian process with mean zero and covariance $\sigma_Z^2 R(t)$. Here, the first term represents a linear regression model, and the second term represents the departure from the assumed linear model, which allows us to interpolate between the observed sites. For $t_i = \{t_{i1},\ldots t_{id}\}$ and $t_j = \{t_{j1},\ldots t_{jd}\}$, the covariance between $Z(t_i)$ and $Z(t_j)$ is given by

$$Cov(t_i, t_j) = V(t_i, t_j) = \sigma_Z^2 R(t_i, t_j), \tag{2}$$

where σ_Z^2 is the process variance of $Z(\cdot)$ and $R(t_i,t_j)$ is a correlation function. Our choice is obtained from the Gaussian correlation family denoted by [8];

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

where θ_k's are non-negative parameters. We define $v'(t_0)$ and $f'(t_0)$ by

$$\begin{aligned} v'(t_0) &= [V(t_0, t_1), \ldots, V(t_0, t_n)], \\ f'(t_0) &= [f_1(t_0), \ldots, f_p(t_0)]. \end{aligned} \tag{4}$$

Here, $v'(t_0)$ is a correlation vector between observed sites and a prediction site t_0, and $f'(t_0)$ is a design vector of t_0.

If the correlation function $R(\cdot,\cdot)$ is known, the best linear unbiased predictor (BLUP) of $Y(t_0)$, given observation y, is

$$\begin{aligned} \hat{Y}(t_0) &= [v'(t_0), f'(t_0)] \begin{bmatrix} V & F \\ F' & 0 \end{bmatrix}^{-1} \begin{bmatrix} y \\ 0 \end{bmatrix} \\ &= f'(t_0)\hat{\beta} + v'(t_0) V^{-1}(y - F\hat{\beta}), \end{aligned} \tag{5}$$

where $F = [f_j(t_i)]_{1 \le i \le n, 1 \le j \le p}$ is an $n \times p$ design matrix of observed sites and $\hat{\beta}$ is the generalized least squares estimator of β; $\hat{\beta} = (F'V^{-1}F)^{-1}F'V^{-1}y$. V is usually unknown. We thus estimate the hyper-parameters in V via the maximum likelihood estimation (MLE) from the data collected at the deisgn sites. Then it are plugged into (5), which makes (5) become the so-called empirical BLUP of $Y(t_0)$ [8] or the Kriging in geostatistics. We used the package "DiceKriging" [20] of the R program.

The prediction model can be determined differently according to a combination of $\beta's$ and $\theta's$ in (1) and (3). Among many possible combinations of those $\beta's$ and $\theta's$, we consider the following four models as basic ones:

β_0 + common θ_c
β_0 + all different $\theta's$
first order liner model + common θ_c
first order liner model + all different θ's.

Here the "common θ" means that d number of $\theta's$ are forced to be a common θ_c such that $\theta_1 = \theta_2 = \ldots = \theta_d := \theta_c$. These four models are employed in this paper. Of course, other models can be considered, and the models based on variable selection algorithm are also possible [21-23]. We refer the readers to [8] and [9] for more information on GPM and its applications to the design and analysis of computer experiments.

3. Approximated Nonlinear Least Squares

We briefly describe the approximated nonlinear least squares (ALS) method proposed by [1]. The following notations for the computer data and for the real experimental data are used:

d: dimension of input variables $t=(c, x)$ of computer code
q: dimension of unknown parameters c
c: unknown parameters to be estimated (q dimensional)
c_S: input variables of computer model corresponding to unknown parameters c (q dimensional)
n_S, n_E: number of observations in computer simulations and in real experiments
x_S, x_E: independent variables in computer model (d-q dimensional) and in real experiments (d-q dimensional)
$Y(c, x), y_E$: computer response for input variables (c, x) and observations in real experiments.

Here, the subscripts S and E represent the computer simulation and real experiment, respectively. If the computer simulation code explains the real experiment data well, we can approximate y_E using the following model:

$$y_E = Y(c, x_E) + e, \tag{6}$$

where e is assumed to be independent and identically distributed $N(0, \sigma_E^2 I)$, and σ_E^2 is the variance of real experiments.

When the computer code is very time-consuming to execute, it is almost impossible in terms of time to optimize some quantity directly from the code. For this case, the ALS uses a GP model as a statistical metamodel or a surrogate of computer code. The ALS first fits the GP model for the computer data by estimating the GP parameters using the MLE. Then, it treats the fitted model as if it were true computer code. The ALS finds c that minimizes the following approximated residual sum of squares (ARSS) [1];

$$ARSS(c) = \sum_{i=1}^{n_E} \left[y_{E,i} - \hat{Y}(c, x_{E,i})\right]^2, \tag{7}$$

where $\hat{Y}(c, x_E)$ is the EBLUP of $Y(x_0)$, as in Eq (5).

The advantage of this method is that it does not require any additional execution of the computer code to evaluate ARSS (**c**) after the prediction model is built from a computer data set [1]. Because no explicit solution is available to minimize ARSS (**c**), we use the quasi-Newton method in "optim" package of R program.

The ALS estimate, c, is obtained for each of the four models (Model 1 to Model 4). An ensemble estimator of c is a simple average of the c's which are obtained from the ALS based on each of the four models. We expect that this simple ensemble estimator has less variance than the estimator based on each of the four models. This is validated and compared using three test functions in the next section, where the true parameters are known a priori.

4. Test Function Study

In this section, we apply the proposed method to test functions. Three test functions in different conditions were studied for a comparative analysis of the methods. The experimental data with n_E sample size and the computer experiment with n_C were generated by

$$y_E = Y(c^*, x_E) + e, \;\; y_C = Y(c_C, x_C).$$

The three test functions along with the true values (c^*) of parameters c are as follows:

Test function 1: $$\begin{aligned} Y(c,x) &= c_1 + c_1x_1 + c_2x_2, \\ c^* &= (3{,}3), n_E = n_C \\ &= 15, e \sim N(0{,}0.2^2) \end{aligned}$$

Test function 2: $$\begin{aligned} Y(c,x) &= c_1exp(|x_1 + x_2|) \\ &+ c_2(x_3 + 1.2x_4 + 1)/2.5 \\ &+ \; 3c_2 \times cos(x_2 + x_3) + c_3, \\ c^* &= (3{,}3{,}3), n_E = n_C \\ &= 40, e \sim N(0{,}0.1^2) \end{aligned}$$

Test function 3: $$\begin{aligned} Y(c,x) &= t_1 * t_2/t_3 + t_4/t_7 \\ &\quad + t_4/t_7 + c_4 - c_3x_4, \\ \text{where } t_1 &= 1 - exp(-1/(2x_2)), \\ t_2 &= 100c_1x_1^2 + c_2, \\ t_3 &= 100c_2x_1^2 + c_1, \\ t_4 &= 20exp(-c_1/10 + c_3x_4logx_1) \\ &\quad + x_3 - c_2, \\ t_7 &= 100(1 + exp((c_1 + c_3x_4) \\ &\quad + logx_2)), \\ c^* &= (3{,}3{,}3{,}6), n_E = n_C \\ &\quad = 60, e \sim N(0{,}0.1^2) \end{aligned}$$

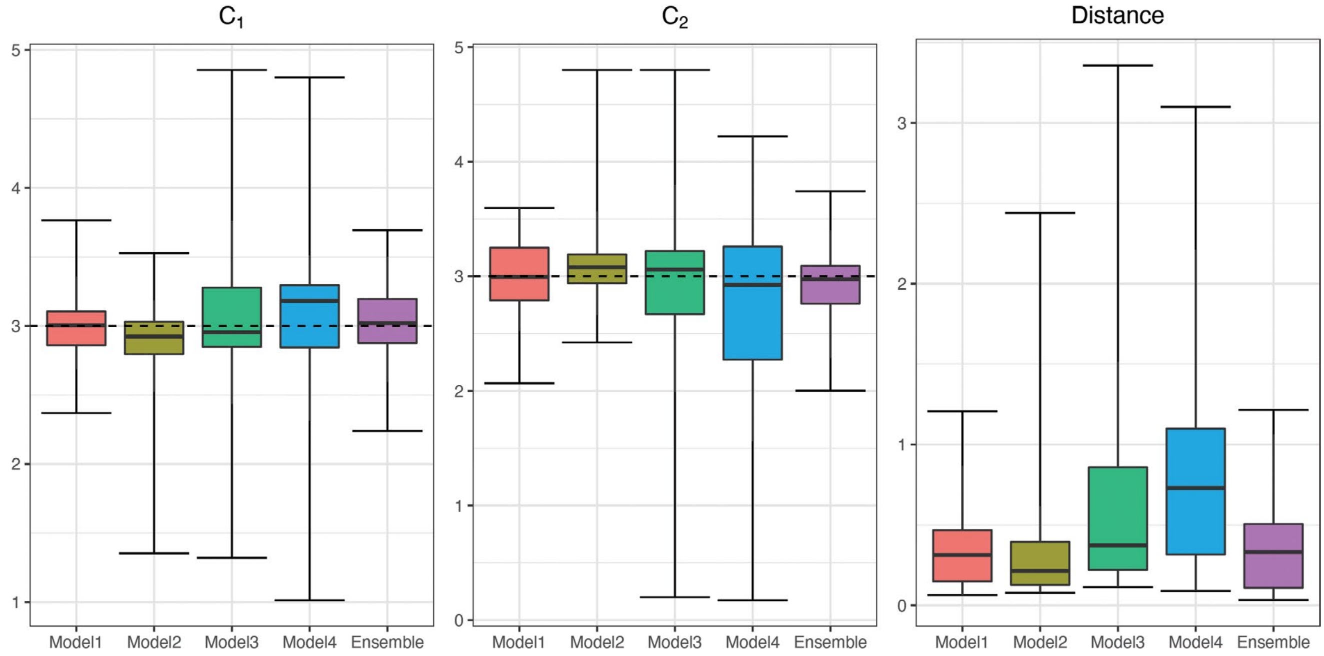

Fig. 1. Boxplots of the estimated calibration parameters which were estimated by the approximated nonlinear least squares methods based on each of four models and ensemble method, and boxplots of the Euclidean distances between true values and estimated values, for test function 1. The dashed horizontal line stands for the true value for each calibration parameter.

The Latin-hypercube designs [24-26] were used to select an independent variable for real experiment (x_E) and an input variable for the computer simulations (c_S, x_S). To address uncertainty in the estimation and the design, 30 different Latin-hypercube designs were employed for each test function. For a comparison analysis, we provide box-plots of 30 different estimates. The Euclidean distance to the true values from the estimates is used to evaluate the accuracy of the methods.

Figs. 1-3 show the results of the performance comparison for each test function. The results show that the ensemble method works better than the ALS based on each of four models, for all three test functions. In test function 1, all ALS methods bsed on four models work well in the sense of narrow boxes in the plot, whereas the wiskers of Models 3 and 4 are long. The ensemble method has narrow box and whisker, similar to the best one (Model 1). In test function 2, the performance of ALS method for c_2 and c_3 are not good, whereas it is good for c_1. The ensemble method still works better than ALS based on each of four models. This findings for test function 2 are also applied similarly to test function 3. In test functions 2 and 3, it seems that Models 3 and 4 work better than Models 1 and 2, respectively, based on the Euclidean distance. It needs further study for justification and generalization.

5. More Calibration Methods

5.1 A generalized ALS

The differences between the observations of experiments and the responses of computer simulations often do not have equal variance or correlated. In that case, a generalized least squares method instead of the ordinary least squares method is usually employed in a linear model theory. Moreover, one potential drawback of the ALS method is that it does not account for uncertainty in the approximation of the computer response by the fitted GP model. To address the above two problems in the calibration study, Lee and Park [6] considered a generalized approximated nonlinear least squares estimation (GALS) method that takes into account the uncertainty due to the approximation and the covariance matrix of residuals.

The GALS method finds c which minimizes the following generalized ARSS ($\mathbf{c}$):

$$GARSS(c) = [y_E - \hat{Y}(T_E)]K^{-1}[y_E - \hat{Y}(T_E)], \tag{8}$$

where $T_E = \{t_{E,1}, t_{E,2}, \ldots, t_{E,n_E}\}'$ is an $n_E \times d$ matrix for $t_{E,i}{}' = (c, x_{E,i})$, and $n_E \times n_E$ matrix K is the covariance matrix of residual $y_E - \hat{Y}(T_E)$ given by

$$K = E[(y_E - \hat{Y}(T_E))(y_E - \hat{Y}(T_E))']. \tag{9}$$

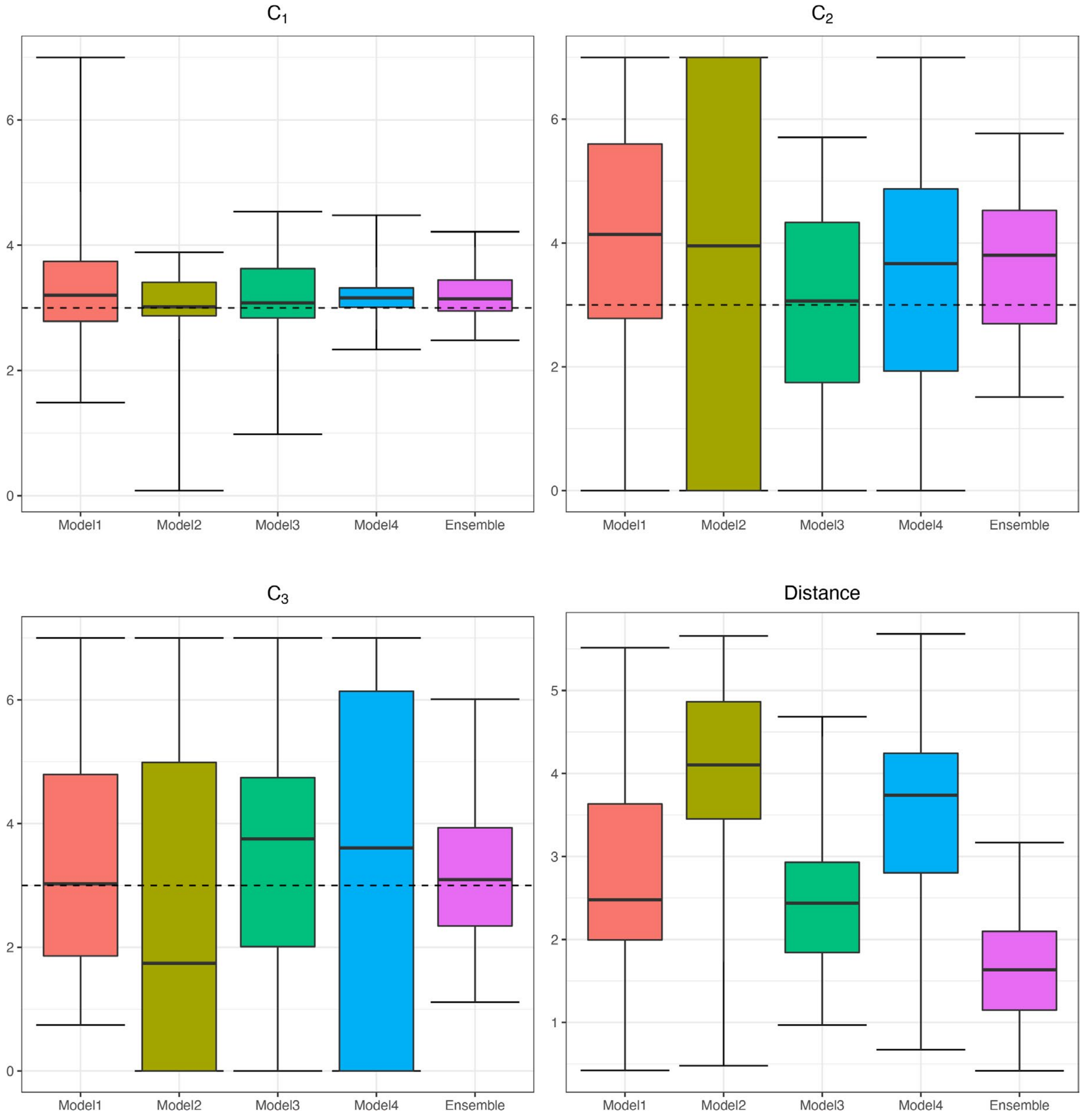

Fig. 2. Same as Fig. 1 but for test function 2.

Through calculations presented in [6], K is written as follows:

$$\sigma_Z^{-2}K = R(T_E,T_E) - (U,F_E)\begin{bmatrix} V_S & F_S \\ F'_S & 0 \end{bmatrix}^{-1}\begin{pmatrix} U' \\ F'_E \end{pmatrix} + \gamma_E I, \tag{10}$$

where $\gamma_E = \sigma_E^2/\sigma_Z^2$, F_S is an $n_S \times p$ design matrix of computer simulations, and F_E is an $n_E \times p$ design matrix for real experiments, respectively. See [6] for more details.

For the calculation of equation (10), the quantities $R\,(T_E, T_E)$, U, V_S, σ_Z^2, and γ_E have to be specified. In this study, the parameters θ's and $\hat{\gamma}_S^2$ are inserted by the MLE calculated from the computer data using each of the four models.

5.2 Iterative version of ALS

Another defect of the ALS is that the surrogate GP is built only once based on the computer data and is not updated thereafter. To solve this difficulty, Seo et al. [4] introduced an iterative version of the ALS. They call it 'max-min' algorithm. The steps of this algorithm are given in the following;

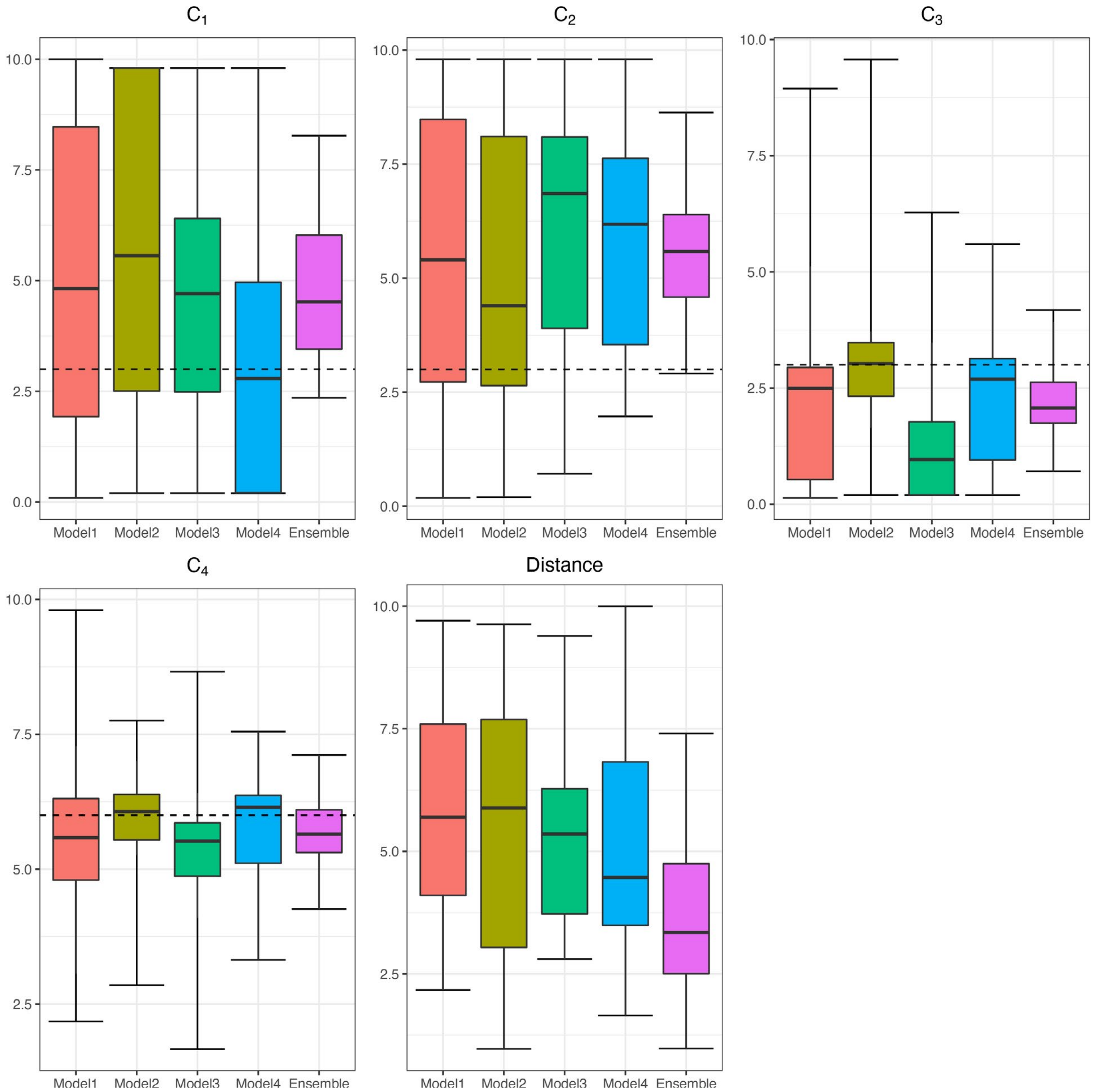

Fig. 3. Same as Fig. 1 but for test function 3.

- **Step 1:** Build a metamodel using MLE for the computer simulation data only.
- **Step 2:** Finds c, which minimizes the ARSS ($\mathbf{c}$) in eq (7) with the estimates from Step 1.
- **Step 3:** Build a metamodel using MLE for the combinded computer data and real experiment data with the fixed parameter c as the obtained value in the previous step.
- **Step 4:** Finds c which minimizes the ARSS ($\mathbf{c}$) in eq (7) with the estimates from Step 3.
- **Step 5:** Repeat Step 3-Step 4 until the stopping rule is satisfied.

In each iteration of Steps 3 and 4, the combined data and the parameters in a GP model are updated, so we expect it positively affect the estimation of c. The advantage of the max-min algorithm is that both of computer data and real experiment data are employed for building a GP model. Thus, the uncertainty in the approximation Y using the metamodel gets smaller than those of the ALS, because the ALS builds a surrogate using computer data only. This method improves the accuracy of calibration and obtains more stable result [4]. As a modification of the above algorithm, one can use the combined data in Steps 1 and 2, which may improve the performance.

5.3 Likelihood-based tuning methods

Given a computer code and a Gaussian process model for y, an unified statistical approach is available. We have the likelihood for all the parameters, including the universal constants τ; the error term parameter $\sigma_{\epsilon E}^2 2$; and the GP parameters $\underline{\beta}$, θ, and σ^2. Thus, all parameters can be estimated by the maximum likelihood method. We refer to this method as the full MLE [1]. It is done for each GP model. The -2 times concentrated log likelihood function (except for constants) of all parameters for the combined data with $\underline{\beta_B}$ and $\hat{\sigma}_B^2$ plugged in is

$$\begin{aligned} &-2\ log\ L(\boldsymbol{\tau}, \eta_B;\ y_B, X_B) \\ &= n_B\ log\ \hat{\sigma}_B^2\ +\ log\ |V_B|, \end{aligned} \tag{11}$$

where

$$\begin{aligned} \hat{\sigma}_B^2 &= (\underline{y_B} - F_B\underline{\beta_B})^t V_B^{-1}(\underline{y_B} - F_B\underline{\beta_B})/n_B, \\ \underline{\beta_B}\ &=\ (F_B{}^t V_B^{-1} F_B)^{-1} F_B{}^t V_B^{-1} \underline{y_B}, \end{aligned} \tag{12}$$

where $\eta_B = (\underline{\theta_B}, \underline{\beta_B}, \sigma_B^2, \gamma_E 2)$, and $\gamma_E 2 = \sigma_{\epsilon E}^2 2/\sigma^2$. Here, the subscript 'B' stands for the combined one of both computer and experimental data.

Some other methods based on the likelihood function were also proposed by [1]. One method is the so-called "SMLE" (Separated MLE), which maximizes the conditional likelihood function of experimental data given the computer data. Here, the parameters $\underline{\beta}$, $\underline{\theta}$, and σ^2 are estimated by maximizing the marginal likelihood for the computer data only. Then these estimates are plugged into the conditional likelihood for the experimental data given the computer data. This is then maximized with respect to γ_E and $\boldsymbol{\tau}$ for the calibration.

One advantage of the likelihood-based methods is that it can simultaneously utilize both computer and experimental data to estimate $\boldsymbol{\tau}$, whereas the ALS uses computer data only. The SMLE is found to be better than the full MLE [1]. These likelihood-based approaches enrich the tuning methods. A comparison of the max-min with the SMLE is provided in [4].

6. Summary and Discussion

To address potential drwabacks of the approximated nonlinear least squares (ALS) calibration method using a Gaussian process surrogate to complex simulator, we considered a simple ensemble method which averages the respective estimates from four different models. Three test functions in different conditions are examined for a comparative analysis of the four models and the proposed method. Based on the test function study, we found that the ensemble method based on four models provided better results than the ALS based on each model.

As a second part of this paper, we reviewed some calibration methods including a generalized ALS, an iterative version of ALS, and likelihood-based method. The generalized ALS constructs the inverse of the covariance matrix of residuals and minimizes the generalized squared residuals with respect to the tuning parameters [6]. In an iterative version of ALS, the parameters are re-estimated and updated by the maximum likelihood estimation (MLE) and the ALS repeatedly until convergence [4]. In the likelihood-based method, the tuning parameters are estimated by the MLE method using both computer and experimental data together.

As an ensemble, an weighted average other than the simple average of $\hat{c}$ would be better for the unbiasedness and efficiency. Then, the next issue is how to assign the weights for each model. We have not tried this approach in this study, but it is the future task.

The estimates $\hat{c}$ may vary according to the selected surrogate; hence, the model selection is very important in calibration. In this study, we only use the simple four models (Model1 to Model 4) for the test function study. If the best model selected by the variable selection method was used, the result might be better. To circumbent this problem, we considered an ensemble estimator in this paper, which showed better accuracy and stability than the estimators based on each model.

The disadvantage of the GALS and the max-min algorithm is that those are computationally more complex than the ALS, because the GALS and the max-min need to optimize more complex functions and needs to estimate more parameters for the covariance matrix K. Nonetheless, the GALS and the max-min may provide a better calibration as well as a better emulator of the computer code than the ALS method can do. Likelihood-based methods can be extended to an iterative version. One extension is the method using EM (expectation and maximization) algorithm [27].

In the literature, calibration is often performed within a framework where the simulation predictions suffer from a systematic bias or discrepancy for any value of parameters. This reflects the view that the mathematical equations underlying the code should not be considered as a perfect model of the real world [2,19,28]. Even if this framework is more realistic, it is outside the scope of this paper. Thus our presentation is focused on a statistical model which does not include the code discrepancy [1]. However, it would be possible to generalize our framework to the inexact simulation model [6].

There are basic limitations with calibrating simulator to real-world data regarding the experimental design. We found that the performance of calibrations is significantly dependent on the designs for both the computer experiments and the physical experiments. Some authors including [29,30] explored this topic. We agree that a sequential approach is practically useful, as in [17,14,13]. Further research on relevant designs under sequential calibration methods would be helpful [4].

Acknowledgements

The authors would like to thank the reviewers and the Editor for helpful comments and suggestions, which have considerably improved the presentation of this paper. This work was supported by Chonnam National University, 2018 (No.2017-2862). We are grateful to Drs. YA Seo and Y Lee for helpful discussion and R programming.

References

1. Cox D, Park JS, Singer CE. A statistical method for tuning a computer code to a data base. Comput Stat Data An 2001;37: 77-92.
2. Kennedy MC, O'Hagan A. Bayesian calibration of computer models. J R Stat Soc B 2001;63:425-464.
3. Han G, Santner TJ, Rawlinson JJ. Simultaneous determination of tuning and calibration parameters for computer experiments. Technometrics 2009;51:464-474.
4. Seo YA, Lee Y, Park JS. Iterative method for tuning complex simulation code. Commun Stat Simulat 2020; Online First. https://doi.org/10.1080/03610918.2020.1728317
5. Trucano TG, Swiler LP, Igusa T, Oberkampf WL, Pilch M. Calibration, validation, and sensitivity analysis: What's what. Reliab Eng Syst Safe 2006;91:1331-1357.
6. Lee Y, Park JS. 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
7. Sacks J, Welch W, Mitchell T, Wynn H. Design and analysis computer experiment (with discussion). Stat Sci 1989;4:409-435.
8. Santner TJ, Williams B, Notz W. The design and analysis of computer experiments, Second Edition. New York: Springer-Verlag; 2018.
9. Gramacy RB. Surrogates: Gaussian process modeling, design, and optimization for the applied sciences. Boca Raton (FL): CRC Press; 2020.
10. Gramacy R, Bingham D, Hollowa J, Grosskopf M, Kuranz CC, Rutter E, et al. Calibrating a large computer experiment simulating radiative shock hydrodynamics. Ann Appl Stat 2015;9: 1141-1168.
11. Wong R, Storlie C, Lee T. A frequentist approach to computer model calibration. J R Stat Soc B 2017;79:635-648.
12. Goh J, Bingham D, Holloway JP, Grosskopf MJ, Kuranz CC, Rutter E. Prediction and computer model calibration using outputs from multifidelity simulators. Technometrics 2013;55:501-512.
13. Damblin, G, Barbillon P, Keller M, Pasanisi A, Parent E. Adaptive numerical designs for the calibration of computer codes. SIAM/ASA J Uncer Quant 2018;6:151-179.
14. Kumar A. Sequential tuning of complex computer models. J Stat Comput Sim 2015;85:393-404.
15. Plumlee M. Bayesian calibration of inexact computer models. J Am Stat Assoc 2017;112:1274-1285.
16. Tuo R, Wu CFJ. Prediction based on the Kennedy-O'Hagan calibration model: asymptotic consistency and other properties. Stat Sinica 2018;28:743-759.
17. Pratola MT, Sain SR, Bingham D, Wiltberger M, Riglerd EJ. Fast sequential computer model calibration of large nonstationary spatial-temporal processes. Technometrics 2013;55:232-242.
18. Gu M, Wang L. Scaled Gaussian stochstic process for computer model calibration and prediction. SIAM/ASA J Uncer Quant 2018;6:1555-1583.
19. Higdon D, Gattiker J, Williams B, Rightley M. Computer model calibration using high dimensional output. J Am Stat Assoc 2008;103:570-583.
20. Roustant O, Ginsbourger D, Deville Y. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and pptimization. J Stat Softw 2012;51.
21. Welch WJ, Buck RJ, Sacks J, Wynn HP, Mitchell TJ, Morris MD. Screening, predicting, and computer experiments. Technometrics 1992;34:15-25.
22. Marrel A, Iooss B, Drope FV, Volkova E. An efficient methodology for modeling complex computer codes with Gaussian processes. Comput Stat Data An 2008;52:4731-4744.
23. Lee Y, Park JS. Model selection algorithm in Gaussian process

regression for computer experiments. Commun Stat Appl Methods 2017;24:383-396

24. Morris M, Mitchell T. Exploratory designs for computational experiments. J Stat Plan Infer 1995;43:381-402.
25. Park JS. Optimal Latin-hypercube designs for computer experiments. J Stat Plan Infer 1994;39;95-111.
26. Carnell R. lhs: Latin Hypercube Samples. R package version 0. 16. 2018.
27. Seo YA, Park JS. An expectation-maximization algorithm for calibrating complex simulation code using Gaussian process emulator. Manuscript submitted 2020.
28. Hong J, Kim TY, Park JS. Multivariate bias correction for climate simulation data, with application to precipitation extremes in Korea. QBS 2019;38:121-130.
29. 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.
30. Beck J, Guillas S. Sequential design with mutual information for computer experiments (MICE): Emulation of a tsunami model. SIAM/ASA J Uncer Quant 2016;4:739-766.