Special Proceeding of the 22nd Annual Conference of SSCA held at Savitribai Phule Pune University, Pune, during January 02–04, 2020; pp 85–101

Statistical Modelling and Calibration of a Dynamic Computer Simulator

Pritam Ranjan
Operations Management and Quantitative Techniques Area
Indian Institute of Management Indore, M.P., India 453556

Received: 05 July 2020; Revised: 13 July 2020; Accepted: 15 July 2020

Abstract

Over the last two decades, the advancement in computing power has led to a significant rise in the usage of computer simulators (or simulation models) for real-life applications where data collection via physical observation or experimentation is infeasible or too expensive. For complex real-life processes, realistic accurate simulation models are also time-consuming to run, and subsequently, statistical surrogate models are often used to emulate the simulator outputs. In this article, we will discuss a surrogate model for emulating dynamic simulator outputs (*i.e.*, a simulator which yields time series response).

For a scalar-valued deterministic simulator, Gaussian process model was first introduced as a statistical surrogate, which is till date the most popular emulator in the computer experiment literature. For a dynamic simulator, this article presents a singular value decomposition (SVD) based Gaussian process (GP) model for the emulation. We will also present an efficient approach for estimating the inverse solution from a dynamic computer model, where the objective is to find the optimal set of inputs that produces outputs matching the target response as close as possible. This is also referred to as the calibration of the computer simulation model. The performance of this innovative approach will be demonstrated via several simulated examples and a real-life application.

Key words: Computer experiments; Expected improvement; Gaussian process regression; Inverse problem; Saddlepoint approximation.

1. Introduction

When the physical processes are too expensive to observe and experiment with, mathematical models are often used to mimic the underlying phenomena. These mathematical models are coded/implemented via some software programming language like C, C++, Java, R, Python, *etc.* The experimentation with such computer codes are referred to as computer experiments. There is a plethora of real-life examples of computer simulators that range from pharmaceutical industry, aviation sector, cosmology, manufacturing, agriculture, re-

Correponding Author: Pritam Ranjan
Email: pritamr@iimidr.ac.in

newable energy, *etc*. We start with briefly presenting a few real-life applications of computer simulators.

Application 1. The data collection for the impact assessment of a car crash due to different types of collision on passengers sitting in the front and back row is really expensive. Though crash testing is a must in some countries, it is not done everywhere and certainly not thoroughly. A popular cheaper alternative is to use simulation models for detailed experimentation. Figure 1 depicts a car-crash experimentation process.

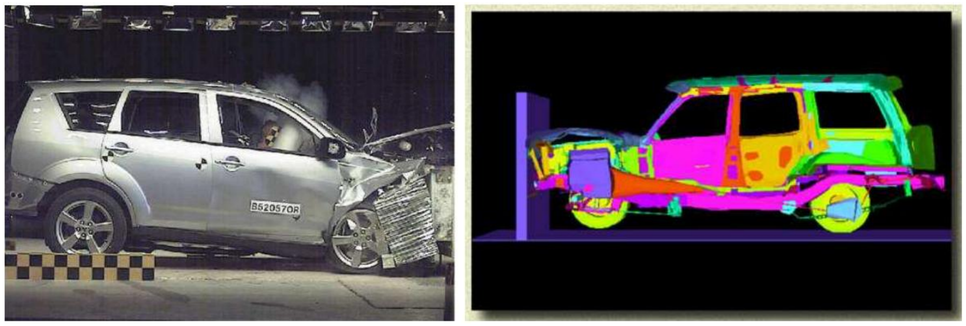


Figure 1: Impact assessment of a car-crash via both physical and computer experiments.

Application 2. Harnessing tidal energy by putting in-stream turbines is an exciting and yet a challenging problem for a variety of researchers around the globe. The Bay of Fundy, located between New Brunswick and Nova Scotia, Canada is world famous for its high tides. In some regions of the Bay of Fundy, the difference in the water level between high tide and low tide can be as much as 17 meters (see Figure 2). The incredible energy in these tides has significant potential for extracting tidal power (Karsten et al., 2008).

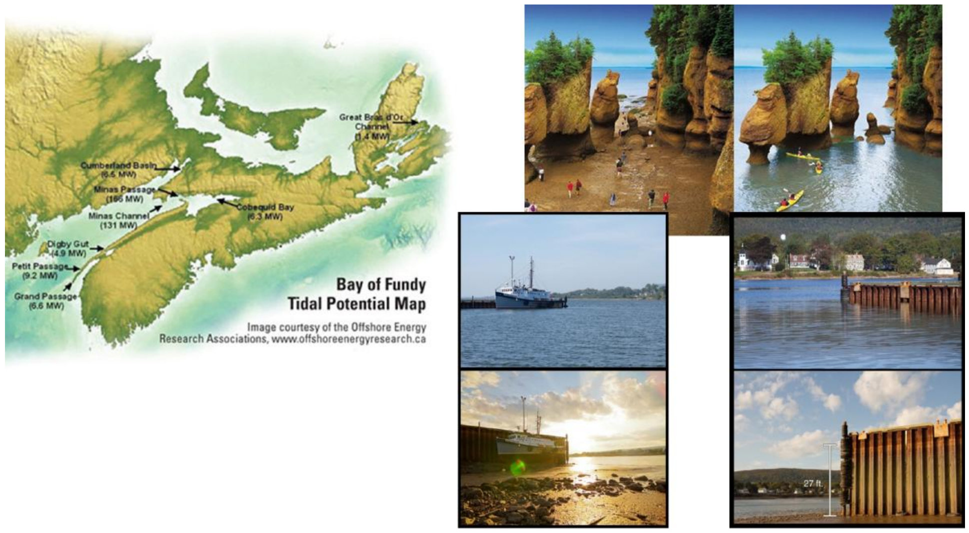

Great Bras d'Or Channel (1.4 MW)
Cumberland Basin (6.5 MW)
Minas Passage (166 MW)
Cobequid Bay (6.3 MW)
Minas Channel (131 MW)
Digby Gut (4.9 MW)
Petit Passage (9.2 MW)
Grand Passage (6.6 MW)
Bay of Fundy Tidal Potential Map
Image courtesy of the Offshore Energy Research Associations, www.offshoreenergyresearch.ca
27 ft

Figure 2: The map of Nova Scotia focussing on the Bay of Fundy. Hopewell Rocks and a port near Wolfville with high tide and low tide.

Preliminary analyses revealed that one can safely extract gigawatts of power by putting a host of in-stream tidal turbines in the Minas Passage - a narrow passage in the Bay of Fundy where the tidal currents are strongest (see Figure 3). One of the interesting research questions is where exactly should the turbines be placed. Identification of the optimal locations cannot be done via physical experimentation, and we must rely of good computer model based experiments. Currently, researchers have been studying different aspects of the problem using versions of the Finite-Volume Coastal Ocean Model (FVCOM). Details of FVCOM can be found in Chen et al. (2006).

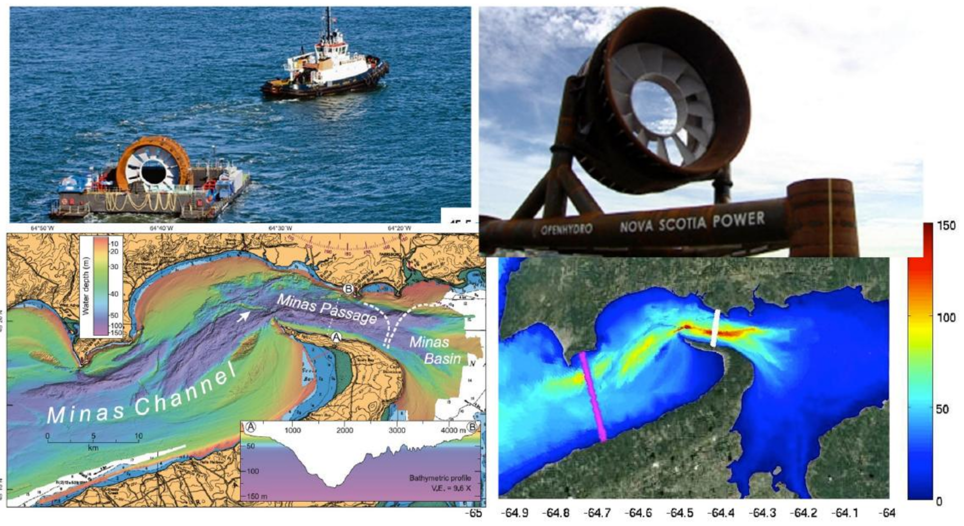


Figure 3: Turbine placement in the Minas Passage.

Application 3. European red mites (ERM) infests on apple leaves and diminish the quality of crop, which inflicts heavy financial loss in the apple industry (see Figure 4). Therefore, the monitoring and subsequent intervention of ERM population dynamics is of vital importance for apple orchards management.

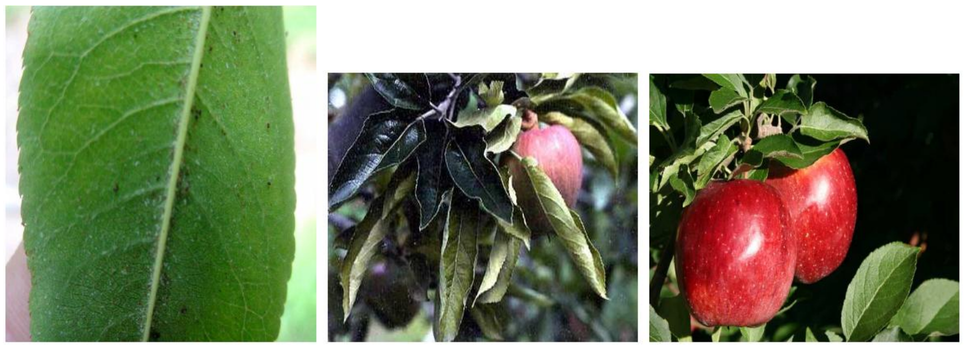


Figure 4: European Red Mites (ERM) infesting on apple leaves.

It is worth noting that data collection for different stages of ERM population, eggs, juveniles and adults, would require counting the number of mites, on randomly sampled orchards, trees, branches and leaves, using a magnifying glass over a period of time. This is undoubtedly expensive (see black solid curve in Figure 5 for a field data). Alternatively, one can use a computer simulator to mimic the population growth of the three stages. The two-delay blowfly (TDB) model (Teismann et al. (2009)) simulates ERM population dynamics under predator-prey interactions via numerically solving the Nicholson's blowfly differential equation (Gurney et al. (1980)). One of the prime objectives here includes the calibration of this simulator to mimic the reality as much as possible. The TDB model takes eleven input variables (*e.g.*, death rates for different stages, fecundity, hatching time, survival rates, and so on) and returns the time series (at 28 time points) of ERM population evolution at three stages, *i.e.*, eggs, juveniles and adults (see Ranjan et al. (2016) for details). Figure 5 shows the model output at five randomly chosen input points.

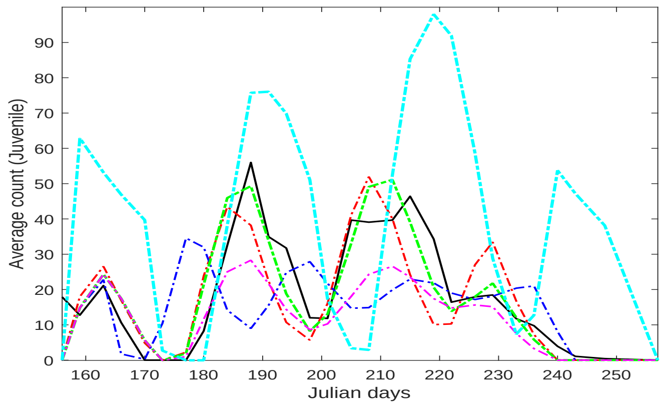


Figure 5: TDB outputs for a few randomly chosen inputs (coloured curved), and the field data (in solid black).

In this article, we first review the popular statistical surrogate called the GP model for emulating the simulator response (Sacks et al., 1989). Subsequently, we discuss the singular value decomposition (SVD) based GP surrogate for emulating the dynamic computer simulator outputs (Higdon et al., 2008; Zhang et al., 2018). Then we focus on the inverse problem for dynamic computer models - which was the key objective of our motivating application from the apple farming industry. This part of the methodology requires a quick recap of the popular expected improvement (EI) based sequential design approach for calibrating the deterministic, expensive to evaluate, scalar-valued simulator outputs (Jones et al., 1998; Ranjan et al., 2008; Bingham et al., 2014), and then present a generalization for dynamic simulators (Zhang et al., 2019). Finally, we compare the performance of this innovative method with the naive approach for several test functions and the TDB model.

2. Statistical Surrogates

The choice of statistical metamodel varies with the characteristics of the computer simulator. For instance, if the output is scalar/vector/functional, the process is stationary/non-stationary, deterministic/stochastic, input space is convex/non-convex, inputs are continuous/discrete, *etc.* We first present the most popular surrogate model called GP model for scalar-valued deterministic simulator, and then outline an extension of this GP model for dynamic simulators.

2.1. Scalar-valued simulator

Let the training data consist of d-dimensional input and 1-dimensional output, denoted by $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ and $y_i = y(x_i)$, respectively. Then, the GP model is written as

$$y_i = \mu + z(x_i), \quad i = 1, 2, \ldots, n, \tag{1}$$

where μ is the overall mean, and $\{z(x), x \in [0,1]^d\} \sim GP(0, \sigma_z^2 R(,))$ with $E(z(x)) = 0$, $Var(z(x)) = \sigma_z^2$, and $Cov(z(x_i), z(x_j)) = \sigma_z^2 R(x_i, x_j)$ where $R(,)$ is a positive definite correlation function. That is, $Y = (y_1, y_2, \ldots, y_n)^T \sim MVN(\mu 1_n, \sigma_z^2 R_n)$, where 1_n is an $n \times 1$ vector of all 1's, and R_n is an $n \times n$ correlation matrix with (i,j)-th element given by $R(x_i, x_j)$ (see Sacks et al. (1989); Santner et al. (2003); Rasmussen and Williams (2006) for more details).

The model described in (1) is typically fitted by either maximizing the likelihood or via Bayesian algorithms like Markov chain Monte Carlo (MCMC). As a result, the predicted response $\hat{y}(x_0)$ is the same as the conditional mean:

$$E(y(x_0)|Y) = \mu + r(x_0)^T R_n^{-1}(Y - 1_n\mu), \tag{2}$$

and the associate prediction uncertainty estimate (denoted by $s^2(x_0)$) can be quantified by the conditional variance:

$$Var(y(x_0)|Y) = \sigma_z^2(1 - r^T(x_0) R_n^{-1} r(x_0)). \tag{3}$$

The most crucial component of such a GP model is the spatial correlation structure, $R(,)$, which dictates the 'smoothness' of the interpolator that passes through the observations. By definition, any positive definite correlation structure would suffice, but the most popular choice is the power-exponential correlation family given by

$$R(x_i, x_j) = \prod_{k=1}^{d} \exp\{-\theta_k |x_{ik} - x_{jk}|^{p_k}\}, \tag{4}$$

where θ_k and p_k controls the wiggliness of the surrogate in the k-th coordinate. A special case with $p_k = 2$ for all $k = 1, 2, ..., d$, represents the most popular Gaussian correlation also known as radial basis kernel in Machine Learning literature.

The parameter estimation for μ, σ_z^2 and θ is often a computationally intensive optimization problem. There are a number of R packages that can provide the GP model fitting, for example, **mlegp**, **GPfit**, **DiceKriging**, **tgp**, **RobustGaSP** and **SAVE** (Dancik, 2013; MacDonald et al., 2015; Roustant et al., 2012; Gramacy, 2007; Gu et al., 2016; Palomo et al., 2015). These R packages are somewhat different in terms of computational efficiency and stability. For the reason of stability, we use the R package **GPfit** in this article. The GP model can be fitted using the following code:

```
GPmodel = GPfit::GP_fit(X, Y, corr = list(type="exponential", power=2))
```

The GPfit object `GPmodel` contains the parameter estimates, which can be further passed on for generating the predictions along with uncertainty estimates on a test set. For example, suppose the simulator output is generated by a one-dimensional test function $f(x) = \log(x + 0.1) + \sin(5\pi x)$, and the input points $\{x_1, ..., x_7\}$ are randomly generated as per a space-filling Latin hypercube design method (McKay et al., 1979). Then, Figure 6 shows the fitted surrogate, prediction uncertainty and the true simulator response curves.

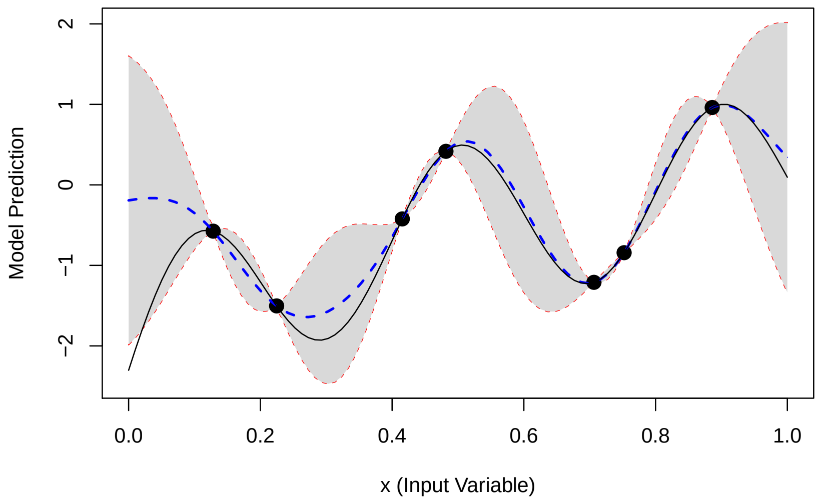


Figure 6: GP regression: The response is generated using the one-dimensional test function $f(x) = \log(x + 0.1) + \sin(5\pi x)$. black solid curve shows the true simulator response, blue dashed curve presents the fitted surrogate and the gray region depicts the uncertainty band.

2.2. Dynamic simulator

A simulator that produces time-series/functional response is referred to as the dynamic computer model. Experimentation via dynamic simulators arise in various applications, for example, rainfall-runoff model (Conti et al., 2009), vehicle suspension system (Bayarri et al., 2007), and the population growth model for European red mites (Zhang et al., 2018) briefly outlined in Section 1.

The time-series structure in the response makes the emulation substantially more challenging as compared to the standard GP model. Recently, a few attempts have been made in this regard. For example, Liu and West (2009) and Farah et al. (2014) proposed time varying autoregressive (TVAR) models. Another clever approach is to represent the time series outputs as linear combinations of a fixed set of basis such as singular vectors (Higdon et al., 2008) and wavelet basis (Bayarri et al., 2007). Zhang et al. (2018) developed an empirical Bayesian approach for the singular value decomposition (SVD) based methodology and generalized it further for large-scale data. We briefly discuss the basic version of SVD-based GP model by Higdon et al. (2008).

Let the simulator inputs and outputs be stored in the $N \times q$ matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T$, and $L \times N$ matrix and $\boldsymbol{Y} = [\boldsymbol{y}(\boldsymbol{x}_1), \ldots, \boldsymbol{y}(\boldsymbol{x}_N)]$, respectively. Then the SVD of $\boldsymbol{Y}$ gives

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T,$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$ is an $L \times k$ column-orthogonal matrix, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_k)$ is a $k \times k$ diagonal matrix of singular values sorted in decreasing order, $\boldsymbol{V}$ is an $N \times k$ column-orthogonal matrix of right singular vectors, and $k = \min\{N, L\}$. Then the simulator response is modelled as

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{i=1}^{p} c_i(\boldsymbol{x})\boldsymbol{b}_i + \boldsymbol{\epsilon}, \tag{5}$$

where $\boldsymbol{x} \in \mathbb{R}^q$, and $\boldsymbol{b}_i = d_i\boldsymbol{u}_i \in \mathbb{R}^L$, for $i = 1, \ldots, p$ represent the orthogonal basis. The coefficients c_i's in (5) are assumed to be independent Gaussian processes, *i.e.*, $c_i \sim \mathcal{GP}(0, \sigma_i^2 K_i(\cdot, \cdot; \boldsymbol{\theta_i}))$ for $i = 1, \ldots, p$, where K_i's are correlation functions. We use the popular Gaussian correlation (4) for $K(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\theta_i})$. The residual term $\boldsymbol{\epsilon}$ in (5) is assumed to be independent $\mathcal{N}(0, \sigma_L^2)$. The number of significant singular values, p, in (5), is determined empirically by the cumulative percentage criterion $p = \min\{m : (\sum_{i=1}^m d_i)/(\sum_{i=1}^k d_i) > \gamma\}$, where γ is a threshold of the explained variation. With carefully chosen priors, the maximum a posteriori (MAP) method gives closed form predictor along with the uncertainty estimates. We follow the empirical Bayesian implementation by Zhang et al. (2018).

R library called *DynamicGP* (Zhang et al., 2020) provides user-friendly functions for quick usage. The most important function is `svdGP`, and its usage is illustrated as follows:

```
svdGP(design, resp, frac=0.95, nthread=1, clutype="PSOCK", ...)
```

where `design` is the input design matrix, `resp` is the output response matrix, `frac` specifies $\gamma = 95\%$, and `nthread` and `clutype` controls the parallelization of the implementation. See Zhang et al. (2020) for details on the implementation.

Suppose the time-series valued response is generated using the following test function (Forrester et al., 2008) with three-dimensional inputs,

$$f(\boldsymbol{x}, t) = (x_1 t - 2)^2 \sin(x_2 t - x_3), \tag{6}$$

where $\boldsymbol{x} = (x_1, x_2, x_3)^T \in [4, 10] \times [4, 20] \times [1, 7]$, and $t \in [1, 2]$ is on a 200-point equidistant time-grid. Figure 7 illustrates the implementation, by first fitting the svdGP model to a training set of 20 input points randomly generated via maximin Latin hypercube design in the three-dimensional hyper-rectangle $[4, 10] \times [4, 20] \times [1, 7]$, and then predicting the time-series valued simulator output using `svdGP()` function.

From Figure 7, it is clear that the fitted surrogate model predictions are reasonable approximations of the simulator outputs at the design points. We fitted svdGP model using the default settings of *DynamicGP* package. Of course, one can play around with other arguments to obtain better (more accurate) predictions.

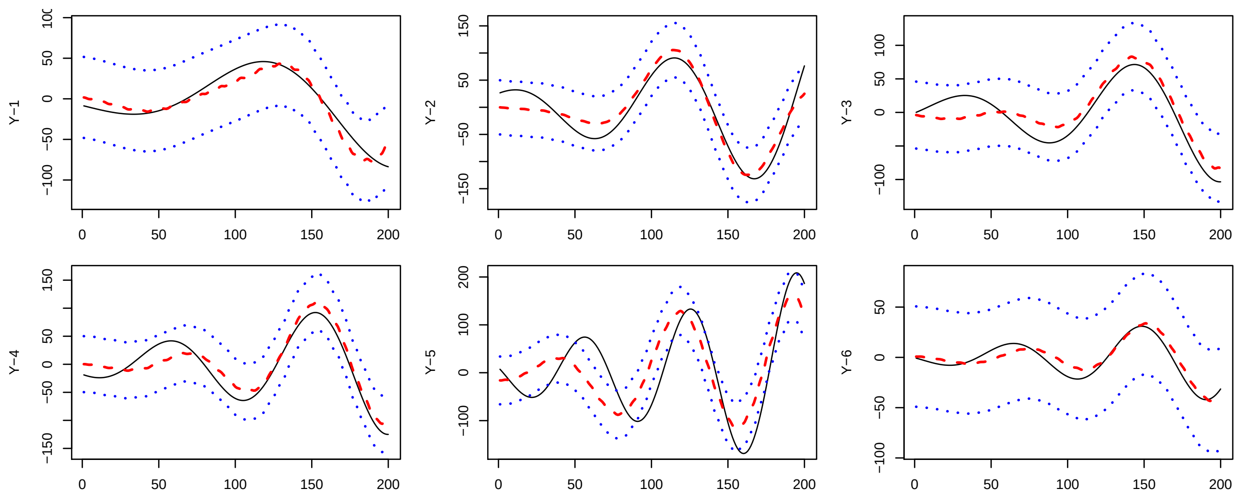


Figure 7: Model prediction for six randomly chosen inputs. Each panel shows the true simulator response (black solid curve), the mean predicted svdGP fit (dashed red curve), and the uncertainty bounds (blue dotted curves).

3. Calibration / Inverse Problem

Calibration of computer simulation models is typically one of the key objectives of computer experiments. For instance, the overall intent of the TDB model application was to ensure that the simulator outputs were as realistic as the observed physical response. This notion of calibration is also referred to as the inverse problem, *i.e.*, the estimation of the set of inputs that yield a pre-specified process value (also called the contour / iso-surface / threshold estimation). Mathematically, the objective is to

$$\text{find } x \in \chi, \;\; s.t. \; y(x) \approx \xi, \quad \text{or, equivalently,} \quad \underset{x\in\chi}{minimize} \; \|\xi - y(x)\|,$$

where $y(x)$ is the simulator response, ξ is pre-defined target to be estimated, and the objective is to estimate $S(\xi) = \{x \in \chi \;:\; y(x) \approx \xi\}$.

Since realistic simulators of complex processes are often computationally expensive, the number of evaluations of the computer simulator is limited which subsequently makes the inverse problem a lot more challenging. A popular efficient strategy in such a situation is to use sequential design approach which starts with an initial design and adds one point or a batch of points at-a-time iteratively until a tolerance based stopping criterion is met or a pre-specified budget is exhausted. The steps are summarized as follows.

- Step 1. Choose an initial design of run size n_0. Let $n = n_0$.
- Step 2. Build a statistical surrogate model with $\{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$.
- Step 3. Choose the next design point $\boldsymbol{x}_{n+1}$ by optimizing a merit-based criterion. Run the simulator at $\boldsymbol{x}_{n+1}$ and obtain y_{n+1}.

- Step 4. Let $n = n + 1$ and repeat Steps 2 and 3 until it reaches the budget (N) or satisfies the stopping criterion.

3.1. Calibration of a scalar-valued simulator

Obviously, $c = 1$ is a terrible choice for the player. For then, she must also choose $k = 1$ toss (since there is no opportunity to toss after capturing one node with the first toss); and she will earn 10, 2, 1, 11 nickels with probability 1/4 each. Therefore, per play she will pay 51 cents; she will earn, on average, $5(10+2+1+11)/4 = 30$ cents; and lose 21 cents—a whopping 41.2% loss!

Suppose that the simulator produces a scalar-valued response, and the objective is to estimate $\xi = a$. Ranjan et al. (2008) developed an expected improvement (EI) criterion (for Step 3) under the GP model in Step 2. The idea is to choose $\boldsymbol{x}_{n+1}$ by maximizing

$$\begin{aligned} \mathrm{E}(I(\boldsymbol{x})) &= \int_{v_1(\boldsymbol{x})}^{v_2(\boldsymbol{x})} [\epsilon^2(\boldsymbol{x}) - (t-a)^2]\phi\left(\frac{t - \hat{y}(\boldsymbol{x})}{s(\boldsymbol{x})}\right) dt \\ &= [\epsilon(\boldsymbol{x})^2 - (\hat{y}(\boldsymbol{x}) - a)^2 - s^2(\boldsymbol{x})](\Phi(u_2) - \Phi(u_1)) + s^2(\boldsymbol{x})(u_2\phi(u_2) - u_1\phi(u_1)) \\ &\qquad + 2(\hat{y}(\boldsymbol{x}) - a)s(\boldsymbol{x})(\phi(u_2) - \phi(u_1)), \end{aligned} \tag{7}$$

where $u_1 = [a - \epsilon(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})]/s(\boldsymbol{x})$, $u_2 = [a + \epsilon(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})]/s(\boldsymbol{x})$, $\phi(\cdot)$ and $\Phi(\cdot)$ are the probability density function and the cumulative distribution function of a standard normal random variable, respectively, and $\epsilon(\boldsymbol{x}) = \alpha s(\boldsymbol{x})$ for a positive constant α. Ranjan et al. (2008) used $\alpha = 1.96$ which is in-sync with 95% confidence interval under the normality assumption of the responses. This criterion is simply obtained by computing the expectation of a carefully designed improvement function $I(\boldsymbol{x}) = \epsilon^2(\boldsymbol{x}) - \min\left\{(y(\boldsymbol{x}) - a)^2, \epsilon^2(\boldsymbol{x})\right\}$ with respect to the predictive distribution $y(\boldsymbol{x}) \sim N(\hat{y}(\boldsymbol{x}), s^2(\boldsymbol{x}))$.

Figure 8 depicts a quick illustration of the contour estimation procedure via the EI criterion for a simulator response generated by a two-dimensional test function.

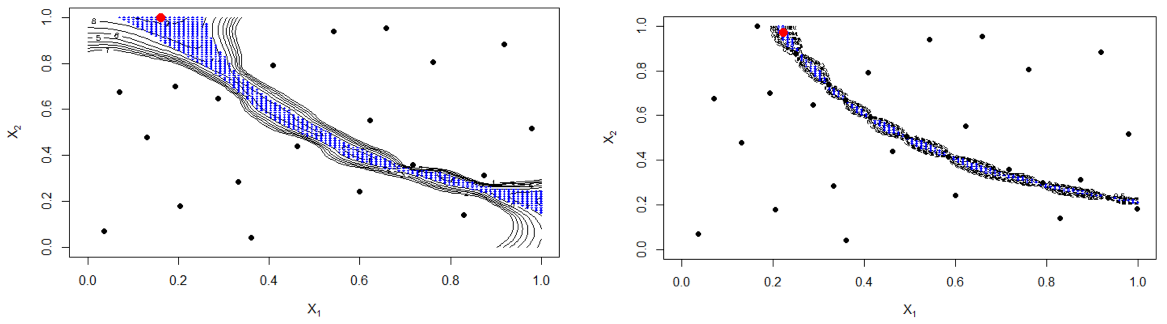


Figure 8: Contour estimation for a two-dimensional test function, $y(x_1, x_2) = (1 + (x_1 + x_2 + 1))(3 + 12x_1x_2)$ at the target $a = 300$, with $n_0 = 20$ and $N = 30$.

3.2. Calibration of a dynamic simulator

When we discuss the estimation of the inverse solution for a dynamic computer model, the problem becomes lot more complicated. Note that the target is now a time series, $\xi = \{\xi_1, ..., \xi_L\}$, and the equality or approximation of $y(\boldsymbol{x})$ and ξ have to be formulated more carefully. For a quick reference, the objective of the TDB model application is to find the set of inputs that would produce model outputs closer to the black solid curve shown in Figure 5. An intuitive definition of the inverse problem is to find

$$x^* = \underset{x \in \chi}{\operatorname{argmin}}\ \delta(x), \qquad \text{where} \ \ \delta(x) = \|\xi - y(x)\|_2^2.$$

3.2.1. A naive approach

Ranjan et al. (2016) suggested treating the mean discrepancy between the model output and the target, $\omega(\boldsymbol{x}) = \sqrt{\delta(\boldsymbol{x})/L}$, as the scalarized simulator output, then use a GP to emulate $\omega(\boldsymbol{x})$, and adopt the EI-criterion in Jones et al. (1998) for finding the global minimum.

Suppose we consider the following three-dimensional test function with inputs $x = (x_1, x_2, x_3) \in [0,1]^3$ to generate the time-series response,

$$g(x,t) = \frac{\sin\left(10\pi t^{(2x_3)}\right)}{(1+2x_1)t} + |t-1|^{(2+4x_2)}, \qquad (8)$$

where the true field data correspond to $x_0 = (0.5, 0.5, 0.5)$. Figure 9 shows the true field data (solid red curve) and a few simulator outputs (blue dotted curves).

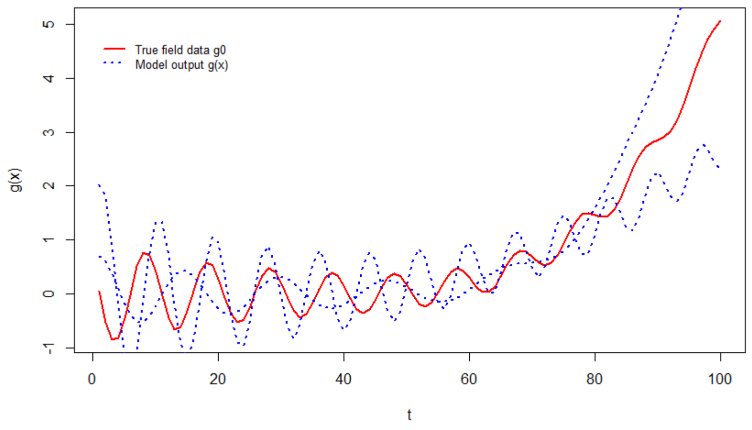
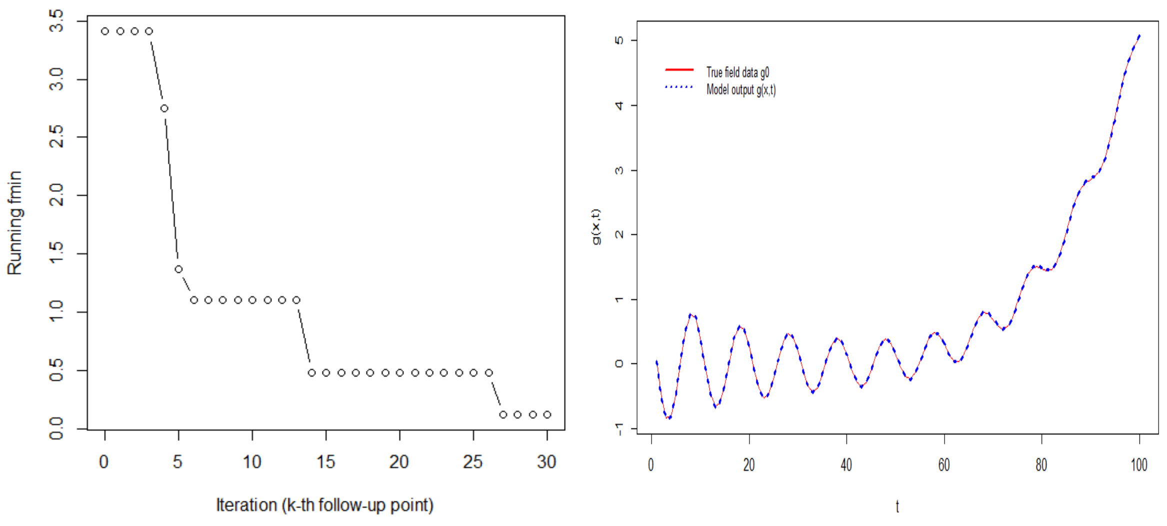


Figure 9: A few computer model outputs and the true data for a three-dimensional test function based dynamic simulator.

Figure 10 shows the implementation of this naive approach for finding the inverse solution using $n_0 = 20$ and $N = 50$.

It turns out that the final estimate of x_{opt} is $(0.4827, 0.4979, 0.4991)$. Though the inverse solution obtained is quite good, there are several theoretical oversights. For instance,



(a) Running estimate of ω_{min} (b) Final inverse solution

Figure 10: Naive approach for solving the inverse problem for a 3-dimensional test function based dynamic simulator.

no efforts are made in Ranjan et al. (2016) for modeling the dynamic computer simulators.

3.2.2. Zhang et al. (2019) approach

Given that the main objective is to efficiently minimize $\delta(\boldsymbol{x}) = \|\boldsymbol{\xi} - \boldsymbol{y}(\boldsymbol{x})\|_2^2$, Zhang et al. (2019) follows the sequential design framework in Jones et al. (1998) for finding the global minimum, and define the improvement function as

$$I(\boldsymbol{x}) = \Big(\delta_{\min} - \delta(\boldsymbol{x})\Big)_+, \tag{9}$$

where $(u)_+ = \max\{0, u\}$ for $u \in \mathbb{R}$ and $\delta_{\min} = \min\{\delta(\boldsymbol{x}_i), i = 1, 2, ..., n\}$. The corresponding EI criterion would look like

$$E[I(\boldsymbol{x})] = E\Big[\Big(\delta_{\min} - \delta(\boldsymbol{x})\Big)_+\Big|\boldsymbol{Y}\Big], \tag{10}$$

where the expectation is taken with respect to the predictive distribution of $\delta(\boldsymbol{x})$ given $\boldsymbol{Y}$. The distribution of $\delta(\boldsymbol{x})$ is not GP, which differentiates this EI with the standard minimization problem in Jones et al. (1998).

The SVD-based GP surrogate model in Section 2.2 presents the distribution of $y(\boldsymbol{x})$ given $\boldsymbol{Y}$, but not the distribution of $\delta(\boldsymbol{x})$ given $\boldsymbol{Y}$. Zhang et al. (2019) used the saddlepoint approximation method to evaluate (10). The method begins by finding a solution of the derivative equation,

$$\kappa_\delta^{(1)}(s) = \delta_{\min}, \tag{11}$$

where $\kappa_\delta^{(1)}(s)$ is the first order derivative of the cumulant generating function of $\delta(\boldsymbol{x})$ with respect to s. Of course there is no closed form solution of (11)), and they used Broyden's method (Broyden, 1965) implemented in the *R* package **nleqslv** (Hasselman, 2010) for numerically solving (11). Let s_0 be the solution of (11), then, for $s_0 > 0, s_0 < 0, s_0 = 0$, Zhang et al. (2019) derived three expressions of saEI (saddlepoint approximation of the expected improvement) which require numerical calculations. The *R* package **DynamicGP** (Zhang et al., 2020) contains built-in function called `saEI` which facilitates easy computation of this criterion for choosing follow-up design points.

4. Examples and Applications

In this section, we present a few examples to illustrate the performance comparison of saEI approach with other competitors like the naive method outlined in Section 3.2.1. We used a realistic example (TDB model) and some test functions for generating dynamic computer simulator outputs. The performance is measured via $\|x^* - \hat{x^*}\|$ and

$$D_\xi = \frac{\|\xi - y(\hat{x}^*)\|_2^2}{\|\xi - \bar{\xi}1_L\|_2^2},$$

where x^* is the truth corresponding to the target response ξ, $\hat{x^*}$ is the estimated optimal inverse solution, $\bar{\xi} = \sum_{i=1}^{L} \xi_{t_i}/L$, and 1_L is an L-dimensional vector of ones.

Example 4 of Zhang et al. (2019): Suppose the outputs of the dynamic simulator obey

$$y_t(\boldsymbol{x}) = \exp(3x_1 t + t)\cos(6x_2 t + 2t - 8x_3 - 6), \tag{12}$$

where $\boldsymbol{x} = (x_1, x_2, x_3)^T \in [0,1]^3$ and $t \in [0,1]$ is on a 200-point equidistant time-grid (Harari and Steinberg, 2014). The input producing the target (or, equivalently, the field observation) is randomly generated as $\boldsymbol{x}^* = [0.522, 0.950, 0.427]^T$. Here, the initial design is of size $n_0 = 18$, and $N - n_0 = 36$ points were chosen sequentially one at-a-time as per the individual design criterion (*e.g.*, using $saEI(\boldsymbol{x})$). Figure 11 summarizes the simulation results over 50 replications.

Example 5 of Zhang et al. (2019): The environmental model by Bliznyuk et al. (2008) simulates a pollutant spill at two locations (0 and L) caused by a chemical accident. The simulator outputs are generated using the following model that captures concentration at space-time point (s,t),

$$\begin{aligned} y_t(\boldsymbol{x}) &= C(s,t;M,D,L,\tau) \\ &= \frac{M}{\sqrt{Dt}}\exp\left(\frac{-s^2}{4Dt}\right) + \frac{M}{\sqrt{D(t-\tau)}}\exp\left(-\frac{(s-L)^2}{4D(t-\tau)}\right) I(\tau < t), \end{aligned} \tag{13}$$

where $\boldsymbol{x} = (M, D, L, \tau, s)^T$, M denotes the mass of pollutant spilled at each location, D is the diffusion rate in the chemical channel, and 0 and τ are the time of the two spills. The input domain is $\boldsymbol{x} \in [7,13] \times [0.02, 0.12] \times [0.01, 3] \times [30.01, 30.304] \times [0,3]$, and $t \in [35.3, 95]$ lies on a regular 200-point equidistant time-grid. In this example, the randomly chosen input that produces the field observation is $\boldsymbol{x}^* = [9.640, 0.059, 1.445, 30.277, 2.520]^T$. Zhang et al. (2019) used a 30-point initial design and 60 follow-up points to estimate the inverse solution. Figure 12 summarizes the results of 50 simulations.

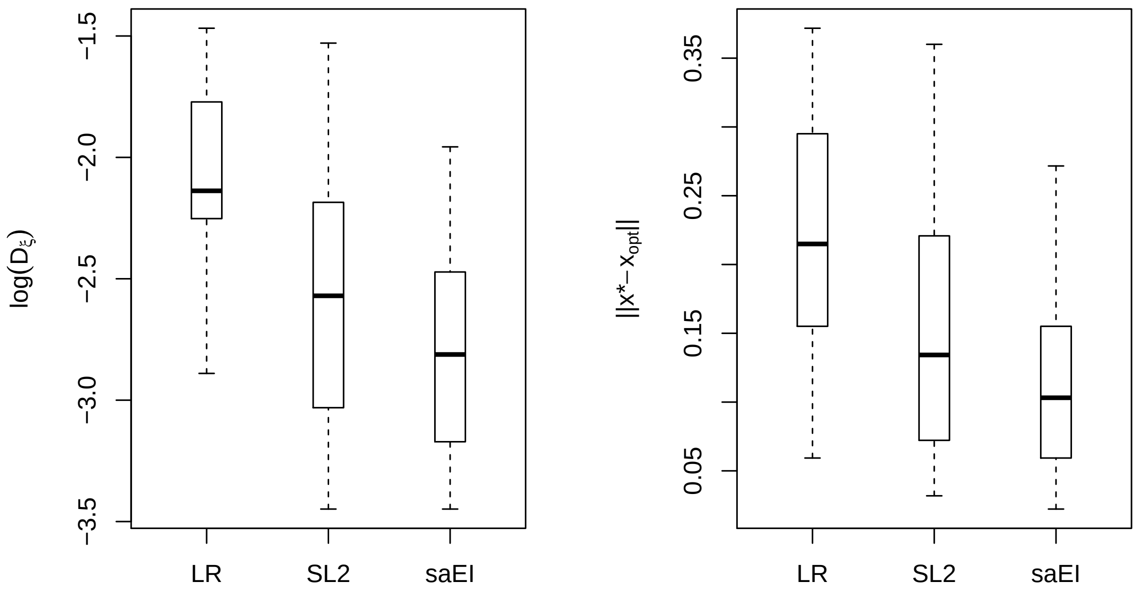


Figure 11: Performance comparison of saEI with SL2 (discussed in Section 3.2.1) and LR method (Pratola et al., 2013). The dynamic responses are generated using a test function from Harari and Steinberg (2014). See Example 4 of Zhang et al. (2019) for details.

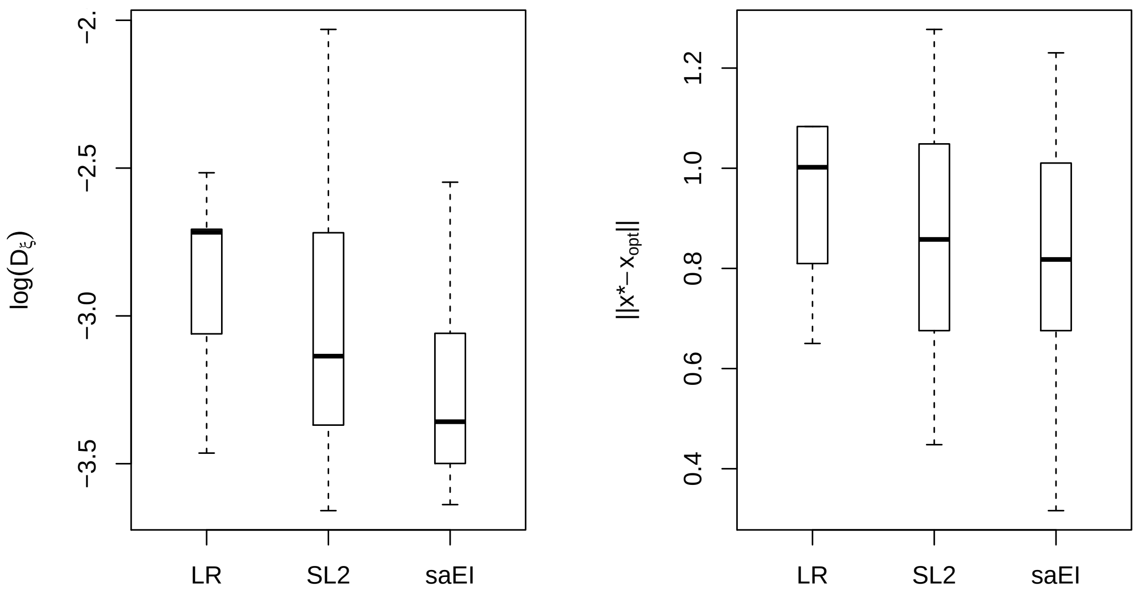


Figure 12: Performance comparison of saEI with SL2 (discussed in Section 3.2.1) and LR method (Pratola et al., 2013). The dynamic responses are generated using a test function from Bliznyuk et al. (2008). See Example 5 of Zhang et al. (2019) for details.

TDB example (Section 4.3 of Zhang et al. (2019)): The target response corresponds to the average count of juvenile population over a 28-day sample during 156-th to 257-th

Julian days (the period during which the apple farming takes place in the Annapolis Valley, Canada). The data was collected twice a week at regular intervals. The version of TDB model used here assumed the following six input variables:

- μ_4 – adult death rate,
- β – maximum fecundity (eggs laid per day),
- ν – non-linear crowding parameter,
- τ_1 – first delay - hatching time of summer eggs,
- τ_2 – second delay - time to maturation of recently hatched eggs,
- Season – average number of days on which adults switch to laying winter eggs.

The performance comparison of the three methods in solving the inverse problem are illustrated in Figure 13. Zhang et al. (2019) used $n_0 = 36$ and $n_{new} = 72$ for finding the inverse solution.

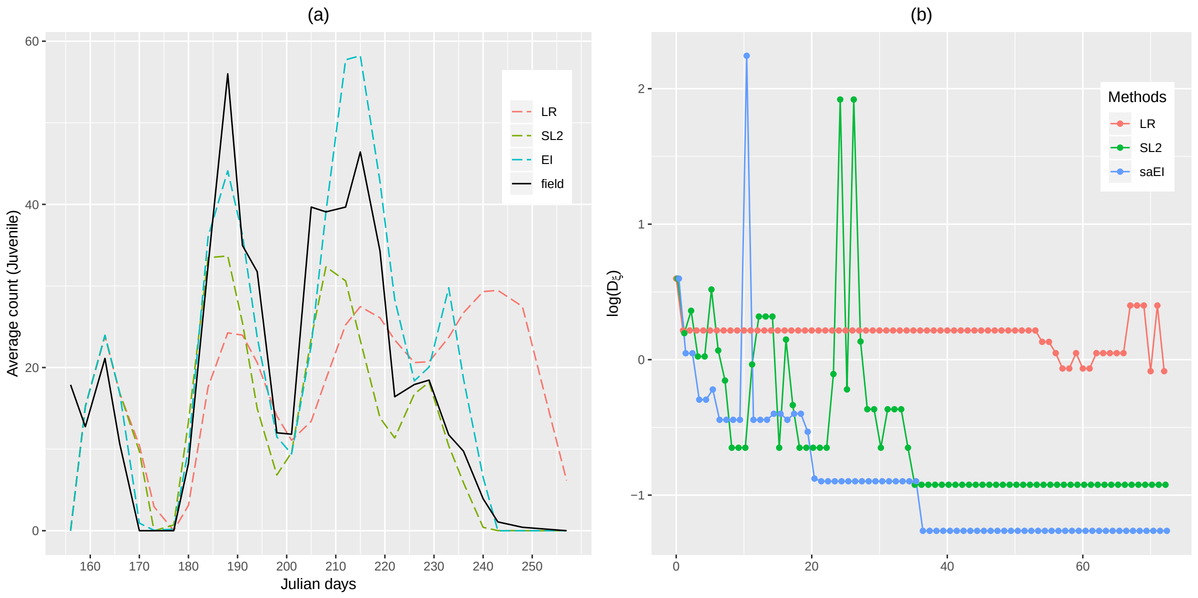


Figure 13: TDB example (Section 4.3 of Zhang et al. (2019)): (a) The field observation of the juvenile ERM population evolution and the located simulator outputs that match the field observation produced by the three sequential design methods. (b) The traces of the $\log(D_\xi)$ criterion values of the 72 iterations for the three methods.

On average, the saEI approach clearly outperforms both the competitors (LR by Pratola et al. (2013) and SL2 by Ranjan et al. (2016)) at least with respect to the examples applications considered here.

5. Concluding Remarks

In this article, we presented the popular statistical metamodel called the GP model for emulating scalar-valued deterministic simulator outputs. This was further generalized to SVD-based GP model for efficient emulation of the dynamic computer simulator response. We also discussed the inverse problem for both scalar and dynamic response simulators that are expensive to evaluate. Several test functions and real-life examples are presented to demonstrate the performance of the methodologies. We also highlight the freely available R libraries for easy implementation of these methodologies. This is particularly helpful for practitioners and young researchers in this area.

There are numerous active research problems in this area such as uncertainty quantification, design and analysis of spatio-temporal simulator data, stochastic simulator data, and in particular how to analyze BIG data coming from computer simulation models.

Acknowledgments This work is primarily based on the Ph.D. thesis of Ru Zhang co-supervised by me and Dr. Chunfang Devon Lin from Queens University, Canada. The key papers published from the thesis are Zhang et al. (2018) and Zhang et al. (2019). I would also like to thank Dr. Vinod Gupta for inviting me to this wonderful conference.

References

Bayarri, M., Berger, J., Cafeo, J., Garcia-Donato, G., Liu, F., Palomo, J., Parthasarathy, R., Paulo, R., Sacks, J. and Walsh, D. (2007). Computer model validation with functional output. *The Annals of Statistics*, **35**(**5**), 1874–1906.

Bingham, D., Ranjan, P. and Welch, W. J. (2014). Sequential design of computer experiments for optimization, estimating contours, and related objectives. In J. F. Lawless (Ed.), *Statistics in Action: A Canadian Outlook* (pp. 109–124). Boca Raton, FL: Chapman & Hall/CRC.

Bliznyuk, N., Ruppert, D., Shoemaker, C., Regis, R., Wild, S. and Mugunthan, P. (2008). Bayesian calibration and uncertainty analysis for computationally expensive models using optimization and radial basis function approximation. *Journal of Computational and Graphical Statistics*, **17**(**2**), 270–294.

Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, **19**(**92**), 577–593.

Chen, C., Beardsley, R. C. and Cowles, G. (2006). An unstructured grid, finite-volume coastal ocean model (fvcom) system. *Oceanography*, **19**, 78–89.

Conti, S., Gosling, J. P., Oakley, J. E. and O'Hagan, A. (2009). Gaussian process emulation of dynamic computer codes. *Biometrika*, **96**(**3**), 663–676.

Dancik, G. M. (2013). *mlegp: maximum likelihood estimates of Gaussian processes.* R package version 3.1.4.

Farah, M., Birrell, P., Conti, S. and Angelis, D. D. (2014). Bayesian emulation and calibration of a dynamic epidemic model for a/h1n1 influenza. *Journal of the American Statistical Association*, **109**(**508**), 1398–1411.

Forrester, A., Sobester, A. and Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide.* John Wiley & Sons.

Gramacy, R. B. (2007). tgp: an R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models. *Journal of Statistical Software*, **19**(**9**), 1–46.

Gu, M., Palomo, J. and Berger., J. O. (2016). *RobustGaSP: Robust Gaussian Stochastic Process Emulation.* R package version 0.5.7.

Gurney, W., Blythe, S. and Nisbet, R. (1980). Nicholson's blowflies revisited. *Nature*, **287**, 17–21.

Harari, O. and Steinberg, D. M. (2014). Convex combination of Gaussian processes for Bayesian analysis of deterministic computer experiments. *Technometrics*, **56**(**4**), 443–454.

Hasselman, B. (2010). *nleqslv: Solve Systems of Non linear Equations, 2010.* R package version 3.3.2.

Higdon, D., Gattiker, J., Williams, B. and Rightley, M. (2008). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, **103**(**482**), 570–583.

Jones, D. R., Schonlau, M. and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, **13**(**4**), 455–492.

Karsten, R., McMillan, J., Lickley, M. and Haynes, R. (2008). Assessment of tidal current energy for the minas passage, bay of fundy. In *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* (pp. 493–507).

Liu, F. and West, M. (2009). A dynamic modelling strategy for Bayesian computer model emulation. *Bayesian Analysis*, **4**(**2**), 393–411.

MacDonald, B., Ranjan, P., Chipman, H., et al. (2015). GPfit: an R package for fitting a Gaussian process model to deterministic simulator outputs. *Journal of Statistical Software*, **64**(**12**), 1–23.

McKay, M. D., Beckman, R. J. and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **42**(**1**), 55–61.

Palomo, J., Paulo, R. and Garcia-Donato, G. (2015). Save: An r package for the statistical analysis ofcomputer models. *Journal of Statistical Software*, **64**(**13**), 1–23.

Pratola, M. T., Sain, S. R., Bingham, D., Wiltberger, M. and Rigler, E. J. (2013). Fast sequential computer model calibration of large nonstationary spatial-temporal processes. *Technometrics*, **55**(**2**), 232–242.

Ranjan, P., Bingham, D. and Michailidis, G. (2008). Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, **50**(**4**), 527–541.

Ranjan, P., Thomas, M., Teismann, H. and Mukhoti, S. (2016). Inverse problem for a time-series valued computer simulator via scalarization. *Open Journal of Statistics*, **6**(**3**), 528–544.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning.* The MIT Press.

Roustant, O., Ginsbourger, D. and Deville, Y. (2012). Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, **51(1)**, 1–55.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, **4(4)**, 409–423.

Santner, T. J., Williams, B. J. and Notz, W. I. (2003). *The design and analysis of computer experiments*. New York: Springer-Verlag.

Teismann, H., Karsten, R., Hammond, R., Hardman, J. and Franklin, J. (2009). On the possibility of counter-productive intervention: the population mean for blowflies models can be an increasing function of the death rate. *Journal of Biological Systems*, **17(4)**, 739–757.

Zhang, R., Lin, C. D. and Ranjan, P. (2018). Local Gaussian process model for large-scale dynamic computer experiments. *Journal of Computational and Graphical Statistics,*, **27**(**4**), 798–807.

Zhang, R., Lin, C. D. and Ranjan, P. (2019). A sequential design approach for calibrating a dynamic population growth model. *SIAM/ASA Journal on Uncertainty Quantification*, **7**(**4**), 1245–1274.

Zhang, R., Lin, C. D. and Ranjan, P. (2020). *DynamicGP: Modelling and Analysis of Dynamic Computer Experiments*. R package version 1.1-6.