Simulation Failure Robust Bayesian Optimization for Estimating Black-Box Model Parameters

Chakrabarty, Ankush; Bortoff, Scott A.; Laughman, Christopher R.

TR2021-128 October 21, 2021

Abstract

Advances in modeling and computation have resulted in high-fidelity digital models capable of simulating the dynamics of a wide range of industrial systems. These models often require calibration, or the estimation of an optimal set of parameters, to reflect a system's observed behavior. While searching over the parameter space is an inevitable part of the calibration process, models are seldom designed to be valid for arbitrarily large parameter spaces. Application of existing black-box calibration methods, therefore, often require repeatedly evaluating a model over a wide range of parameters. For some parameter combinations, the simulations could be unreasonably slow or fail altogether. In general, the shape of subregions in the parameter space that could result in simulation failure is unknown and near-impossible to ascertain analytically. In this paper, we propose a novel failure robust Bayesian optimization (FR-BO) algorithm that learns these failure regions from simulation data and informs a Bayesian optimization algorithm to avoid failure regions while searching for optimal parameters. This results in acceleration of the optimizer's convergence and prevents wastage of time trying to simulate parameters with high failure probabilities. Index Terms—Machine learning; Bayesian optimization; model simulation; digital twin; feasibility analysis; numerical methods; Gaussian processes.

IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2021

Simulation Failure Robust Bayesian Optimization for Estimating Black-Box Model Parameters

Ankush Chakrabarty[†], Scott A. Bortoff, and Christopher R. Laughman[*]

Abstract—**Advances in modeling and computation have resulted in high-fidelity digital models capable of simulating the dynamics of a wide range of industrial systems. These models often require calibration, or the estimation of an optimal set of parameters, to reflect a system's observed behavior. While searching over the parameter space is an inevitable part of the calibration process, models are seldom designed to be valid for arbitrarily large parameter spaces. Application of existing black-box calibration methods, therefore, often require repeatedly evaluating a model over a wide range of parameters. For some parameter combinations, the simulations could be unreasonably slow or fail altogether. In general, the shape of subregions in the parameter space that could result in simulation failure is unknown and near-impossible to ascertain analytically. In this paper, we propose a novel failure robust Bayesian optimization (FR-BO) algorithm that learns these failure regions from simulation data and informs a Bayesian optimization algorithm to avoid failure regions while searching for optimal parameters. This results in acceleration of the optimizer's convergence and prevents wastage of time trying to simulate parameters with high failure probabilities.**

Index Terms—**Machine learning; Bayesian optimization; model simulation; digital twin; feasibility analysis; numerical methods; Gaussian processes.**

I. Introduction

Current trends towards model-based system development and the application of digital twins place an increasing emphasis on the use of modeling and simulation for large-scale systems [1]. One essential step in the development of these technologies involves model calibration, where the goal is to estimate a set of parameters for a simulation model that result in predictions which accurately reflect observed system behavior. Most formulations iteratively solve an optimization problem through repeated simulation [2]–[4], as parameter estimates are gradually adjusted to minimize a model-fitting cost. A widespread assumption is that the model can be simulated forward in time over any duration of interest for any set of parameters within an admissible search domain.

This assumption does not always hold. It is not uncommon for calibration algorithms to attempt a simulation for parameter values that cause the simulation to fail, since, for black-box models, one does not have prior knowledge regarding such a *failure region*. Many existing simulation-oriented models exhibit multi-scale dynamics, significant nonlinearities, and numerically stiff behavior that can result in simulations that take a significant amount of time to run or fail entirely [5]. These challenges are particularly common in building/HVAC dynamics models that seek to describe the temporal behavior of occupied buildings with their associated closed-loop space cooling systems [6], due to their widely separate timescales, hybrid (continuous/discrete) behavior, and nonlinear interactions between physical subsystems. Rather than expend significant effort to identify parameter sets that result in successful simulations, it is common practice to ignore the information conveyed by a failed simulation for a given parameter set and simply re-run the simulation with a different (often arbitrarily selected) set of parameters.

Practical calibration methods are often designed to estimate near-optimal parameters without extensive simulations to avoid expenditure of significant time and resources without a corresponding increase in simulation quality. Recently, Bayesian optimization (BO) [7] has emerged as an effective method for learning parameters based on limited data in a few-shot manner [8]: that is, with markedly fewer evaluations of the cost function (equivalently, model simulations) than population-based methods. Furthermore, Bayesian optimization inherently balances exploration and exploitation and can incorporate non-convex constraints via modified acquisition functions [9], making it a powerful and easy-to-use learner for model calibration.

In this paper, we use the information that results from a failed simulation during the model calibration process to accelerate the convergence of the calibration process and improve the quality of parameter estimates. We thus propose a novel variant of BO called *failure robust Bayesian optimization* (FR-BO), which comprises modules that use simulated data to delineate regions in the parameter space where the system is likely to fail. Subsequently, we design a novel acquisition function, inspired by constrained BO acquisition functions, that allows searching for optimizer candidates that not only fit the data well, but also are unlikely to result in simulation failures.

II. Preliminaries

We denote by

$$y_{0:T} = \mathcal{M}_T(\theta) \tag{1}$$

a general model of a dynamical system, wherein the constant parameters of the model are described by $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$. The admissible set of parameters Θ is assumed to be known. For instance, Θ could denote a set of upper and lower bounds on parameters obtained from archived data or domain knowledge. Since the model is black-box, it is

[†]Corresponding author. Email: chakrabarty@merl.com. Phone: +1 (617) 758-6175.

[*]All authors are affiliated with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

not uncommon for such a range to be purely a guess, and therefore, not tight around the true parameter set. The output vector $y_{0:T} \in \mathbb{R}^{n_y \times T}$ contains all measured outputs from the dynamical system obtained over a time period $[0, T]$. We do not make any assumptions on the underlying structure of the model $\mathcal{M}_T(\theta)$, where simulating $\mathcal{M}_T(\theta)$ forward with a fixed (and admissible) set of parameters θ yields a vector of outputs $y_{0:T} := \begin{bmatrix} y_0 & y_1 & \cdots & y_t & \cdots & y_T \end{bmatrix}$, with each output measurement $y_t \in \mathbb{R}^{n_y}$.

We aim to estimate parameters $\theta^\star$ that minimize a cost based on the modeling error:

$$\varepsilon \triangleq y^\star_{0:T} - \mathcal{M}_T(\theta^\star), \tag{2}$$

where $y^\star_{0:T}$ denotes the measured outputs collected from a real system, and $\mathcal{M}_T(\theta^\star)$ denotes the estimated outputs from the model $\mathcal{M}_T(\theta)$ using the estimated parameters $\theta^\star$. To this end, we propose optimizing a calibration cost function $J(y^\star_{0:T}, \mathcal{M}_T(\theta))$ to yield the optimal parameters

$$\theta^\star = \arg\min_{\theta \in \Theta} J(y^\star_{0:T}, \mathcal{M}_T(\theta)). \tag{3}$$

Recent work has shown that Bayesian optimization (BO) is effective at finding global optima of functions whose gradients are not available and are expensive to evaluate, as is the case in black-box model calibration [8].

While various methods have been proposed for solving (3), most (if not all) these solutions assume that $\mathcal{M}_T(\theta)$ is valid for every $\theta \in \Theta$, which implies that the model $\mathcal{M}_T(\theta)$ can be simulated from the time-span of interest $[0, T]$ for any parameter in the admissible parameter space Θ. Unfortunately, this is not always the case and model simulations can fail to complete in a timespan of interest. Models that exhibit simulation failures do so in some typically unknown *failure region* $\Theta_\mathsf{F} \subset \Theta$ and the failure does not always occur instantaneously. Consequently, data-driven algorithms that have been designed agnostic to simulation failure could potentially continue to compute optimizer candidates that reside in the failure region Θ_F. In such cases, the algorithm could deteriorate in performance and lead to large amounts of computational resources and time being wasted.

Under the critical assumption that $\theta^\star \notin \Theta_\mathsf{F}$, our **objective** is to design a data-driven parameter estimation framework that can learn from simulation failures and incorporate this information to increase the probability of selecting sets of parameters that lead to successful simulations, thereby optimizing (3) without wasting computational resources. To fulfill this objective, we propose a failure-robust Bayesian optimization (FR-BO) approach wherein we first design a probabilistic classifier that can estimate the failure region Θ_F from simulation data obtained by sampling within Θ. Since function evaluations are assumed to be expensive, we employ an entropy-based active learning method to reduce the sample complexity of this step [10]. Once an estimate $\hat{\Theta}_\mathsf{F}$ of Θ_F is obtained, the classifier provides probabilities of simulation failure over the entire parameter space of interest Θ. Failure probabilities can be embedded into a Bayesian optimization framework through a failure-classifier informed acquisition function, ensuring that optimizer candidates are biased to reside outside $\hat{\Theta}_\mathsf{F}$. We posit that if the classifier is well designed, it will suggest optimizer candidates that lie within the set difference $\Theta \setminus \Theta_\mathsf{F}$ with high probability.

Components of the FR-BO framework include:

(1) a *failure classifier* for estimating likelihoods of simulation success and failure on Θ by learning the set Θ_F;
(2) a *parameter-to-cost regressor* for learning a map from parameters θ to the cost J; and
(3) a *failure robust acquisition function* that incorporates probability of simulation failure into the BO framework.

III. Failure Robust Bayesian Optimization

In this section, we describe a way to learn the failure region from data obtained during simulations by posing it as a classification problem. The classifier can be used via active learning to accelerate the Bayesian optimization step, but also to guide where best to simulate the dynamical model $\mathcal{M}_T(\theta)$ to get better estimates of the failure region boundaries. We also explain how to incorporate information from the classifier via a novel acquisition function to promote the selection of parameters outside the learned failure regions.

A. Learning Failure Regions

1) Data collection: Since the admissible parameter search domain Θ is known, one can sample on this space to obtain a training set for learning the failure region subset Θ_F. In the sequel, we will discuss how to select $\theta \in \Theta$ that are most informative (in an information-theoretic sense), but we will assume that such a training dataset is initially available, for instance, obtained by random sampling on Θ. At the j-th iteration of training the failure classifier, the training dataset

$$\mathcal{D}_j = \theta_{[0:j]} \times \ell_{[0:j]} \times J_{[0:j]}$$

comprises a sequence of parameters $\theta_{[0:j]}$, a sequence of corresponding simulation failure labels $\ell_{[0:j]}$, and cost function values $J_{[0:j]}$. Each failure label is denoted $+1$ if there is simulation failure and -1 if not. For failed simulations, we set the corresponding cost function value to some nonsense value, e.g., `NaN`. If the simulation is successful, the cost function yields a real-valued scalar.

2) The failure classifier: We use a scalable variational Gaussian process classifiers [11] as a failure classifier; an advantage of this method is that we can assign a probability of failure to any parameter in the search space. Inducing a distribution rather than deterministic classification outputs enables the proposed failure robust acquisition function, which we will discuss later.

At the j-th iteration, one can utilize the θ and labels ℓ of the dataset $\mathcal{D}_j$ to set a Gaussian process prior at the observed parameter sets. This can be written as $\phi \sim \mathcal{N}(0, K_j)$, where ϕ_j is the prior function using $\mathcal{D}_j$ and

$$K_j = \begin{bmatrix} \mathcal{K}(\theta_0, \theta_0) & \cdots & \mathcal{K}(\theta_0, \theta_j) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\theta_j, \theta_0) & \cdots & \mathcal{K}(\theta_j, \theta_j) \end{bmatrix}, \tag{4}$$

with a user-specified kernel function $\mathcal{K}(\cdot,\cdot)$ such as a squared exponential kernel or a Matern kernel; see [12] for more details about kernel functions.

To perform classification with this prior, one needs to transform the function ϕ through a squashing function such as the cumulative density function of a zero-mean unit-variance normal distribution $\gamma(\cdot) := \mathcal{N}(\cdot|0,1)$, given by $\Gamma(z) = \int_{-\infty}^{z} \gamma(\alpha)\, \mathrm{d}\alpha$. Consequently, a Bernoulli distribution can be used to represent a likelihood function conditioned on the transformed data as follows:

$$\mathcal{B}(\ell_j|\Gamma(\phi_j)) = \Gamma(\phi_j)^{\ell_j} \cdot (1-\Gamma(\phi_j)^{1-\ell_j}).$$

The joint distribution of ℓ and ϕ thus becomes

$$p(\ell,\phi) = \prod_{r=1}^{j} \mathcal{B}\left(\ell_j|\Gamma(\phi_j)\right) \mathcal{N}(0,K_j). \tag{5}$$

Two more distributions are required to optimize hyperparameters and perform inference: the marginal likelihood $\mathsf{PF}(\ell,j)$ and the posterior $p(\phi|\ell,j)$. Both of these distributions require the inversion of the $j\times j$ kernel matrix (4), which incurs cubic complexity and does not scale well to the large values of j that may be required for FR-BO to compute good solutions. We therefore resort to the use of approximation methods that leverage pseudo-inputs, which are more commonly known as inducing points [13].

Inducing points $\tilde{\theta} \in \Theta$ are design variables that are augmented with the latent variables ϕ_j that also respect the Gaussian prior and therefore yield a joint distribution

$$(\phi,\tilde{\phi}) \sim \mathcal{N}\left(0, \begin{bmatrix} K_j & \tilde{K}_{jm} \\ \tilde{K}_{jm}^{\top} & \tilde{K}_m \end{bmatrix}\right).$$

$\tilde{K}_{jm}$ denotes the covariance matrix computed by evaluating the kernel across j data points and m inducing inputs, while $\tilde{K}_m$ denotes the covariance matrix computed by evaluating the kernel on all pairs of the inducing inputs. Exploiting the properties of the Gaussian distribution, one can rewrite the joint distribution of the latent variables and the inducing variables as

$$p(\ell,\phi,\tilde{\phi}) = p(\ell|\phi)p(\phi|\tilde{\phi})p(\tilde{\phi}).$$

To get a variational approximation of the likelihood, the following inequality is used from [11]:

$$\log p(\ell|\tilde{\phi}) \geq \mathsf{E}_{p(\phi|\tilde{\phi})}[\log p(\ell|\phi)].$$

Defining a variational distribution q, we get the well-known variational lower bound

$$\log p(\ell) \geq \mathsf{E}_{q(\phi)}[\log p(\ell|\phi)] - \mathsf{KL}[q(\tilde{\phi})\|p(\tilde{\phi})]. \tag{6}$$

The optimal hyperparameters for the VGPC can be obtained by minimizing the loss function formed by the negative of the right hand side of this inequality using quadrature methods. If we assume $q \sim \mathcal{N}(\tilde{\phi}|\tilde{\mu},\tilde{\Sigma})$, then

$$q(\phi) = \mathcal{N}(L\tilde{\mu}, K_j + L(\tilde{\Sigma}-\tilde{K}_m)L^{\top}), \tag{7}$$

with $L = \tilde{K}_{jm}\tilde{K}_m^{-1}$, which is an $m\times m$ matrix, and eventually $m \ll j$, so this matrix is cheaper to invert, which makes this method scalable.

3) Active learning: For inference at a set of test points $\{\theta_*\}$, we transform those into $\{\phi_*\}$, and the approximate posterior is then given by

$$p(\phi_*|\ell) = \int p(\phi_*|\tilde{\phi})q(\tilde{\phi})\,\mathrm{d}\tilde{\phi},$$

which can be computed in a manner similar to (7). Initially, when the failure region is not estimated well, it is necessary to select informative elements of $\{\theta_*\}$ that can yield good estimates of Θ_{F} without exhaustive sampling. To this end, we propose an active learning strategy wherein the most informative $\theta_j^\star \in \theta_*$ is selected based on the maximum entropy of the posterior distribution, similar to [10]. Since the mean and variance of the posterior $p(\phi_*|\ell)$ is computed for each parameter in θ_*, one can compute the entropy (assuming q is Gaussian) and fix

$$\theta_j^\star = \arg\max_{\theta'\in\theta_*} \frac{1}{2}\log\left(2\pi\mathsf{Var}(\theta')\right).$$

We then evaluate $\theta_j^\star$ by simulating $\mathcal{M}_T(\theta_j^\star)$ to ascertain ℓ_{j+1} and J_{j+1}, which yields the updated dataset $\mathcal{D}_{j+1}$ and the process iterates till a termination criterion such as a maximum number of iterations is achieved. We have found the active learning using VGPC useful for an initial exploration of the parameter space Θ to significantly reduce the number of failed simulations during Bayesian optimization.

B. Classical Bayesian optimization

Classical BO methods assume the presence of one global optimum, and smoothness of the θ to J map. Since J is typically assumed to be continuous, one can leverage the data at the j-th iteration to construct a surrogate GP model of the reward, given by

$$\hat{J}_j := \mathsf{GP}\left(\mu(\theta;\mathcal{D}_j), \sigma(\theta,\theta';\mathcal{D}_j)\right), \tag{8}$$

where $\mu(\cdot)$ is the predictive mean function, and $\sigma(\cdot,\cdot)$ is the predictive variance function, and $\mathcal{D}_j$ containing $\{\theta_{[0:j]}, J_{[0:j]}\}$ is the dataset collected thus far. Typically, the variance is expressed through the use of kernel functions [7].

At the j-th learning iteration, for a new query sample $\theta \in \Theta$, the GP model predicts the mean and variance of the reward to be $\mu(\theta) = k_j(\theta)^{\top}K_j^{-1}J_{0:j}$ and $\sigma(\theta) = \mathcal{K}(\theta,\theta) - k_j(\theta)K_j^{-1}k_j(\theta)^{\top}$, where $k_j(\theta) = \begin{bmatrix}\mathcal{K}(\theta_0,\theta) & \mathcal{K}(\theta_1,\theta) & \cdots & \mathcal{K}(\theta_j,\theta)\end{bmatrix}$, and K_j is defined in (4).

The accuracy of predicted mean and variance is strongly linked to the selection of the kernel and the best (in some sense) set of hyperparameters such as length scales and variance parameters of the kernels and estimated noise. We obtain these hyperparameters by maximizing the log marginal likelihood function (MLL)

$$-\frac{1}{2}\log|K_j| - \frac{1}{2}J(\theta)^{\top}K_j^{-1}J(\theta) - \frac{n_\theta}{2}\log 2\pi.$$

Note that inducing variables as in the VGPC can be used to significantly improve scalability of the GP [14].

In Bayesian optimization, we use the mean and variance of the surrogate model $\hat{J}_j$ in (8) to construct an acquisition function to inform the selection of a θ_j that increases the likelihood of minimizing the current best cost. To this end, we compute the incumbent $\hat{J}_j^\star := \min_{\theta\in\Theta} \mu(\theta; \mathcal{D}_j)$ and use it to define an expected improvement (EI) acquisition function that has the form

$$\mathsf{EI}(\theta, j) = \begin{cases} \sigma(\theta)\gamma(z) + (\hat{J}_j^\star - \mu(\theta))\Gamma(z), & \text{if } \sigma(\theta) > 0, \\ 0 & \text{if } \sigma(\theta) = 0. \end{cases}$$

where $z = \frac{\hat{J}_j^\star - \mu(\theta)}{\sigma(\theta)}$, and $\gamma(\cdot)$, $\Gamma(\cdot)$ are the PDF and the CDF of the zero-mean unit-variance normal distribution, respectively.

In the j-th iteration of learning, we use the data $\mathcal{D}_j$ to construct the EI acquisition function using the surrogate $\hat{J}_j$. Subsequently, we compute the optimizer candidate

$$\theta_{j+1} = \arg\max_{\theta\in\Theta} \mathsf{EI}(\theta, j), \tag{9}$$

which serves as the parameter estimate θ in (1) in the next BO iteration. In practice, other acquisition functions such as the lower confidence bound or entropy search could also be used instead of EI [7].

C. Incorporating failure probabilities

Since the VGPC generates a probabilistic output, we can directly incorporate it into the acquisition function, as proposed in [9]. This yields the failure-robust EI acquisition function

$$\mathsf{FREI}(\theta, j) = \mathsf{EI}(\theta, j) \cdot (1 - \mathsf{PF}(\theta, j)), \tag{10}$$

where $\mathsf{P}F(\theta, j)$ is the likelihood of failure calculated by training the VGPC using data up to the j-th iteration, and then evaluating the likelihood of the VGPC at θ.

If the VGPC algorithm does not find any θ such that $\mathsf{P}(\theta) > 0$, then the acquisition function (10) is zero for every $\theta \in \Theta$ and future candidates are selected randomly until at least one θ is found which allows for a successful simulation. This scenario is rarely seen in practice. A more plausible scenario is that both successful simulations and failure simulations have been observed, and the VGPC has been trained on a non-trivial classification problem. In such a case, the higher the value of $\mathsf{P}(\theta, j)$, the higher the probability that a particular candidate will be selected, so long as its expected improvement is high as well. The multiplicative nature of the components in (10) seeks to ensure that neither one component can outweigh the other, and candidates will be selected only if they are both a candidate for optimization and feasibility (i.e., is expected to yield a successful simulation). Along the same lines as (9), FR-BO selects the next optimizer candidate as follows:

$$\theta_{j+1} = \arg\max_{\theta\in\Theta} \mathsf{FREI}(\theta, j). \tag{11}$$

A key difference with FREI and the constrained expected-improvement acquisition function proposed in [9] is that we estimate likelihood off binary labels ℓ, whereas constrained BO estimates probabilities based on continuous slack variables obtained from the constraints. Both methods are similar in that they use Gaussian process proxies for constraint-modeling, although in our case it is a classifier rather than the regressor proposed in [9].

It is important to note that from an implementation perspective, it may be expensive to retrain a VGPC after the collection of each new data sample. Empirically, we have observed that this is not always necessary: in fact, as long as the VGPC has been trained initially with some data, it can be retrained infrequently. Of course, how frequently the retraining has to occur is problem dependent, although heuristics such as retraining the VGPC when the FR-BO is 'stuck' at a local optimum for a pre-decided number of iterations can be useful.

IV. Results and Discussion

We provide two examples that demonstrate the effectiveness of FR-BO. The first involves parameter estimation for a well-studied stiff nonlinear system of chemical kinetics, and the second is a real-world building model calibration problem. All code was implemented in `GPyTorch` [15], `PyTorch` [16], and `Python 3.9`.

Example: Estimating rate parameters for the Robertson system

We consider the Robertson system

$$\begin{aligned} \dot{x}_1 &= -\theta_1 x_1 + \theta_2 x_2 x_3 & y &= x \\ \dot{x}_2 &= \theta_1 x_2 - \theta_2 x_2 x_3 - \theta_3 x_2^2 & 0 &= x_1 + x_2 + x_3 - 1 \\ \dot{x}_3 &= -\theta_3 x_2^2, \end{aligned}$$

where x_1, x_2, and x_3 are concentrations of chemical species, and the true parameters of the system are $\theta_1 = 4 \times 10^{-2}$, $\theta_2 = 10^4$, and $\theta_3 = 3 \times 10^7$. This system is well known to be highly stiff and has recently been investigated for surrogate modeling [17]. Note that the system of equations is assumed unknown during parameter estimation using FR-BO. We collect ground truth by simulating the Robertson system forward for $T = [0, 20]$ s, after which we partition T into 100 equidistant samples in order to obtain $y_{0:T}^\star$. The admissible parameter space is defined in logarithmic space and is given by $\log_{10} \Theta = \{[-3, -1] \times [2, 12] \times [5, 15]\}z$. To clarify, θ_1 can lie in the interval $[10^{-3}, 10^{-1}]$ and so on. For each θ sampled on Θ, we simulate the Robertson system over T using the Radau IIA solver, which is well-suited to the numerical integration of stiff systems [5]. For each θ, we can then compute $\mathcal{M}_T(\theta)$ and a parameter estimation cost function

$$J(y_{0:T}^\star, \mathcal{M}_T(\theta)) = \log\left(\sum_{t=0}^{T} (y_t^\star - y_t)^\top W (y_t^\star - y_t)\right),$$

where y_t is the output vector obtained from $\mathcal{M}_T(\theta)$ at the t-th time instant. The matrix W is a scaling matrix to ensure that the three output components are of similar magnitudes. We set $W = \mathrm{diag}[1, 5\times 10^4, 1]$. We collect 300 data points using active learning for the VGPC and then run 700 BO iterations, with the VGPC retrained every 100 iterations during BO. Table I describes the hyperparameters used for both the VGPC and the GPR learners needed for FR-BO.

TABLE I
FR-PO IMPLEMENTATION FOR ROBERTSON SYSTEM.

	VGPC	GP
Model	Approximate GP	Exact GP
Kernel	Squared Exponential	Matern-3/2
Inducing Points	Yes	Yes
Likelihood	Bernoulli	Gaussian
Loss Function	Variational ELBO	MLL
Optimizer	Adam	Adam
Learning Rate	0.01	0.05
Training Iters	2000	500

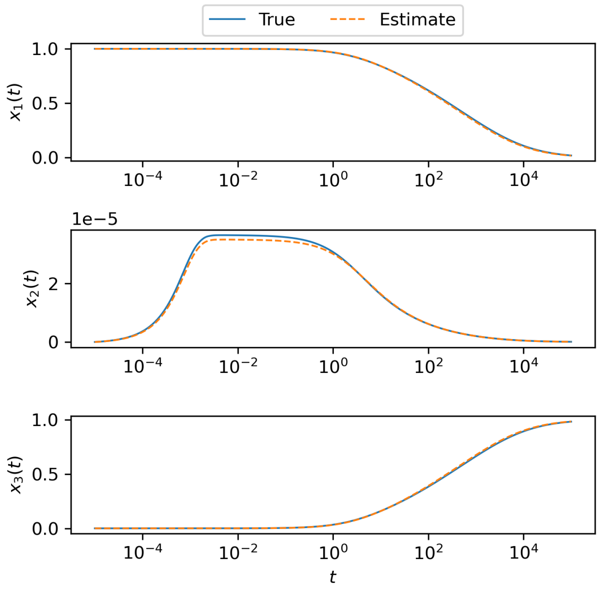

Fig. 1. Robertson system: Comparison of true outputs and estimated outputs on testing scenario $t \in [10^{-5}, 10^5]$.

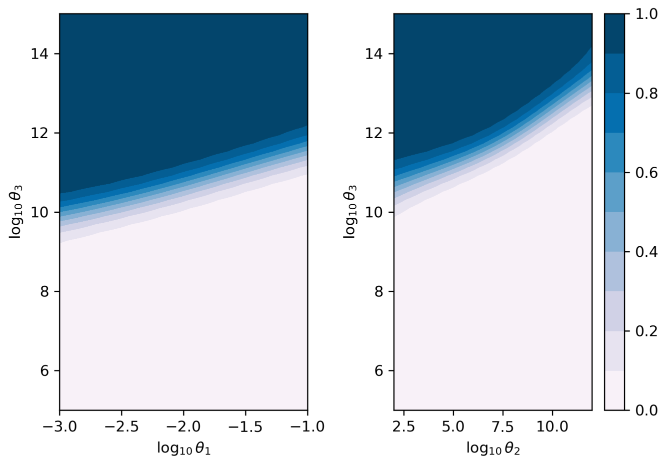

Fig. 2. Robertson system: Slices of the parameter space with classifier estimating probabilities of failure.

After applying FR-BO, the set of parameters that produces the lowest cost is given by $\hat{\theta}_1 = 3.717 \times 10^{-2}$, $\hat{\theta}_2 = 0.887 \times 10^4$, and $\hat{\theta}_3 = 3.03 \times 10^7$, which is satisfactorily close to the true parameters. We validate the effectiveness of the estimated parameters on a time span of $[10^{-5}, 10^5]$ as suggested in [18] to showcase multi-scale dynamics. The similarity of the model outputs and the true outputs is demonstrated in Fig. 1 and the mean squared error between the two trajectories is 9.910×10^{-5}. We also demonstrate two slices of the failure region learned by the VGPC in Fig. 2. The slice on the left is obtained by fixing θ_2 at its estimated value and allowing θ_1 and θ_3 to vary over the subset of Θ, and the slice of the right is obtained by fixing θ_1 and allowing the other two parameters to vary. Both slices clearly indicate that the VGPC has identified a region where the model simulation is likely to exhibit failure; moreover, it is also evident that such a region can have significant curvature. This is evidence that one must resort to advanced nonlinear classifiers like VGPC rather than linear classifiers, even for low-dimensional dynamical systems.

To understand whether FR-BO offers benefits over heuristics that could theoretically provide avenues around simulation failures, we compare the FR-BO algorithm against three other methods. The first is sampling via Monte-Carlo methods, e.g., low-discrepancy sampling. The second method, which we call Zr-BO, runs classical BO and ignores simulation failures; when a candidate θ_j results in a simulation failure, that θ_j is not added to the dataset and the algorithm continues with jitter to ensure it does not get stuck in an infinite loop. The third method, referred to as Hi-BO, replaces the `NaN` costs associated with simulation failures with a high cost function value for θ if it results in a simulation failure. By associating a high cost for those θ values that result in simulation failure, the intuition is that the classical BO algorithm will automatically avoid those areas in future iterations. We maintain parity among the methods as much as possible by allowing identical kernels, hyperparameters, number of iterations, random seeds, and initial dataset obtained by the VGPC.

Fig. 3 shows the best cost function value up to iteration j against the j-th horizontal-axis location over optimization iterations. This figure also illustrates which iterations resulted in a simulation failure via the red vertical lines. The first 300 iterations are identical for all four methods, since that is the initial set of parameters and cost values obtained using the VGPC. After iteration 300, the VGPC is ignored by all algorithms except FR-BO. It is clear from these plots that the FR-BO algorithm outperforms all the competitors in terms of cost decay after 600 iterations and in terms of simulation failures, as it finds a better solution at the 1000-iteration mark and exhibits no simulation failure after the initial VGPC is trained. Conversely, Zr-BO and MCMC sampling both result in large numbers of simulation failures, with Zr-BO showing a large number of failures after iteration 300 as the exploration aspect of Zr-BO keeps trying to search on the edges of Θ in regions where the model is very likely to fail. Of the three competitors, Hi-BO performs relatively well, and certainly reduces the cost over iterations, although there are a significant number of simulation failures. This indicates that the total CPU-time required for running Hi-BO could be far larger than FR-BO if simulation failures do not occur quickly.

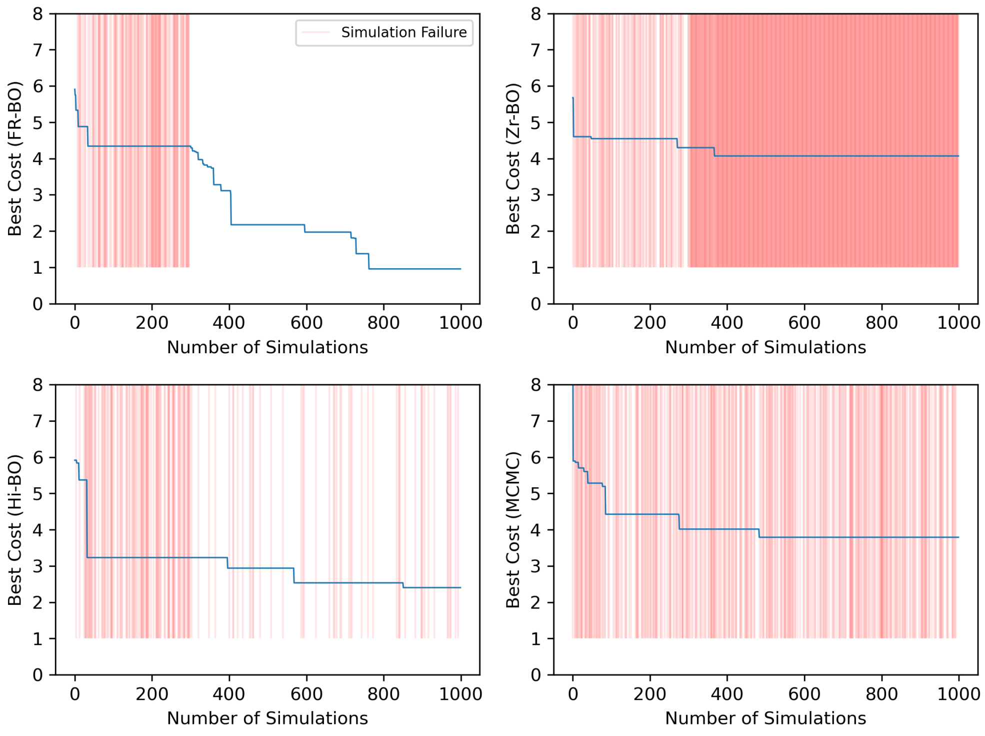

Fig. 3. Robertson system: Comparison of best solution found up to an iteration and failures during Bayesian optimization.

V. Conclusions

Simulation software is dsigned and validated over a limited region of the available parameter space. Thus, regions of validity for model parameters, or regions over which the simulation will fail, are rarely known during downstream design. Calibrating the model for different datasets requires exploring the parameter space and could involve trying parameters that will result in simulation failures. In this paper, we provided a methodology for model calibration using failure-robust Bayesian optimization that involves learning the failure region and embedding that information into the exploitation step via a failure-robust acquisition function, and demonstrated the efficacy of this method on a well-studied numerically stiff system.

References

[1] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[2] J. Ma, B. Huang, and F. Ding, "Iterative identification of hammerstein parameter varying systems with parameter uncertainties based on the variational bayesian approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 3, pp. 1035–1045, 2017.

[3] L. Kovács, M. Siket, I. Rudas, A. Szakál, and G. Eigner, "Discrete lpv based parameter estimation for tidm patients by using dual extended kalman filtering method," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 1390–1395.

[4] M. Wani, F. Hafiz, A. Swain, and A. Ukil, "Estimating thermal parameters of a commercial building: A meta-heuristic approach," *Energy and Buildings*, vol. 231, p. 110537, 2021.

[6] C. Laughman *et al.*, "Modeling and control of radiant, convective, and ventilation systems for multizone residences," in *Proc. of Building Simulation 2019*, 2019, pp. 1956–1963.

[5] E. Hairer and G. Wanner, "Stiff differential equations solved by Radau methods," *Journal of Computational and Applied Mathematics*, vol. 111, no. 1-2, pp. 93–111, 1999.

[7] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. NeurIPS*, 2012, pp. 2951–2959.

[8] A. Chakrabarty, E. Maddalena, H. Qiao, and C. R. Laughman, "Data-driven calibration of joint building and HVAC dynamic models using scalable Bayesian optimization," in *IBPSA Building Simulation Conference*, 2021.

[9] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints." in *Proc. ICML*, vol. 2014, 2014, pp. 937–945.

[10] A. Chakrabarty, C. Danielson, S. Di Cairano, and A. Raghunathan, "Active learning for estimating reachable sets for systems with unknown dynamics," *IEEE Transactions on Cybernetics*, 2020.

[11] J. Hensman, A. G. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," *J. Mach. Learn. Res.*, vol. 38, pp. 351–360, 2015.

[12] C. K. Williams and C. E. Rasmussen, *Gaussian Processes For Machine Learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[13] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Proc. NeurIPS*, vol. 18, pp. 1259–1266, 2006.

[14] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, 2005.

[15] J. R. Gardner *et al.*, "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration," in *Proc. NeurIPS*, 2018, pp. 7587—-7597.

[16] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proc. NeurIPS*, 2019, pp. 8024–8035.

[17] R. Anantharaman, Y. Ma, S. Gowda, C. Laughman, V. Shah, A. Edelman, and C. Rackauckas, "Accelerating simulation of stiff nonlinear systems using continuous-time echo state networks," *arXiv preprint arXiv:2010.04004*, 2020.

[18] S. Kim, W. Ji, S. Deng, and C. Rackauckas, "Stiff neural ordinary differential equations," *arXiv preprint arXiv:2103.15341*, 2021.