Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

CALIBRATION USING EMULATION OF FILTERED SIMULATION RESULTS

Özge Sürer
Matthew Plumlee

Department of Industrial Engineering and Management Sciences
Northwestern University
2145 Sheridan Rd
Evanston, IL 60208, USA

ABSTRACT

Calibration of parameters in simulation models is necessary to develop sharp predictions with quantified uncertainty. A scalable method for calibration involves building an emulator after conducting an experiment on the simulation model. However, when the parameter space is large, meaning the parameters are quite uncertain prior to calibration, much of the parameter space can produce unstable or unrealistic simulator responses that drastically differ from the observed data. One solution to this problem is to simply discard, or filter out, the parameters that gave unreasonable responses and then build an emulator only on the remaining simulator responses. In this article, we demonstrate the key mechanics for an approach that emulates filtered responses but also avoids unstable and incorrect inference. These ideas are illustrated on a real data example of calibrating COVID-19 epidemiological simulation model.

1 INTRODUCTION & BACKGROUND

Simulation models are used to understand real world predictions but they are often burdened with parameters that are either unknown or imprecisely known. Calibration is the process of statistically learning these parameters with requisite uncertainty quantification. During this process, the simulation output is aligned with the real observations and uncertainty is pulled through the process. While one might imagine calibration as a one-shot process, this usage is less common than it once was. Today, simulation calibration is often done regularly because models and data can rapidly be updated and changed. This motivates the need for simulation calibration tools that are automated, fast and reliable.

Bayesian statistical inference is a natural and popular candidate for simulation calibration. There also exist some non-Bayesian ways of handling this brand of calibration, e.g. Yuan et al. (2012) or Plumlee (2019), but in many calibration settings the prior knowledge and the need to quantify the uncertainty make Bayesian inference the best candidate. In a Bayesian calibration paradigm, the computation often uses Markov Chain Monte Carlo (MCMC) for posterior inference (Gelman et al. 2013, Chap. 11). MCMC is a sequential sampling algorithm that can involve millions of simulation model runs to generate approximate samples from the posterior distribution. This requirement can be problematic if running a simulator takes hours or days. An alternative approach is to replace the simulation with a cheaper emulator (Higdon et al. 2008; Plumlee 2017; Wong et al. 2017), which is built by using the simulation runs at chosen parameters to produce a predictive distribution for the simulation output at parameters where the simulation was not run. The emulation often uses a Gaussian process framework to easily pull through the uncertainty in the predictions, see the texts of Santner et al. (2018) and Gramacy (2020). It is then relatively easy to integrate this emulation prediction with calibration in a manner that allows for uncertainty quantification (Higdon et al. 2008; Plumlee 2017; Wong et al. 2017). There are also sequential algorithms that use the emulation to drive collection of new data, e.g. (Wang, Zhang, and Xie 2019) and (Yuan and Ng 2020), but the focus of this article is the analysis of parameters after a large, parallel simulation expedition.

2021 Winter Simulation Conference (WSC) | 978-1-6654-3311-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/WSC52266.2021.9715296

© IEEE 2022. This article is free to access and download, along with rights for full text and data mining, re-use and analysis.

When the initial parameter space is large, or the model is sufficiently sensitive to the parameters, it is possible to get extremely unrealistic results in some regions of the parameter space. In this paper, we present a new approach for the calibration of black-box simulators with unrealistic results that avoids unstable and incorrect inference. We demonstrate our approach using a COVID-19 epidemic simulator as a case study in which disease dynamics are characterized by an enhanced Susceptible-Exposed-Infectious-Removed (SEIR)-style model of disease transmission (Wang et al. 2020). The simulation results (e.g., forecasted daily admissions and census hospitalizations, daily and census ICU hospitalizations, confirmed cases, and deaths) specifically used the COVID-19 differential equation-based simulation model of Yang et al. (2021). Yang et al. proposes a simulation-based-optimization model to decide on whether the community mitigation measures should be enhanced or relaxed to guide public policies throughout the COVID-19 epidemic. Because some of the simulation outcomes yield hospitalizations that diverge from real data, Yang et al. propose a filtering approach to discard the simulator outputs that are unreasonable, meaning they are inconsistent with observed hospitalizations in their model to create a robust COVID-19 alert system.

Filtering, or discarding, unreasonable parameter settings based on simulation results has recently been investigated under the heading of Approximate Bayesian Computation (ABC) (Beaumont et al. 2002; Rutter et al. 2019). In basic ABC, one begins by simulating a massive swath of runs at various parameters. This is also the first step in an emulation-based approach, but typically it is imagined that the sample size is much larger for ABC. The second step in ABC is simply to remove all simulations that are sufficiently far from the observations, measured somewhat arbitrarily by an acceptability function. There is a huge benefit of speed and simplicity in the deployment of ABC. ABC inference is formally proven to agree with Bayesian inference when the acceptability function is sufficiently tight. However, if we choose an acceptability function that is too stringent, then ABC can be wasteful if many simulation outcomes are far from the actual observations. On the other hand, a loose acceptability function may result in including many sub-optimal parameters in the resulting inference, and the conclusions will drift further from the correct posterior. There are many direct advantages of using emulation over ABC, and Baker et al. (2020) describe some key details.

The purpose of this article is to properly marry emulation-based calibration inference with filtration in a generic, statistical learning framework. We will do so by deploying a broad acceptability function, thus expanding the acceptable region, and then building an emulator using data from the simulations that were deemed acceptable. Our broad acceptability function thus discards the unreasonable results from simulation that are not helpful for representing the underlying truth. This is in contrast to ABC where the acceptability function is used to fully determine what results are representative of reality. This process has two key benefits over emulating across the entire parameter space. First, emulators are easier to train and typically more accurate in the reduced space as unreasonable response regions can give unstable and ill-behaved emulation results. The resulting calibration models from the proposed filter-emulate-calibrate procedure is thus more robust than the direct emulate-calibrate procedure. Second, building an emulator with filtered data can be computationally cheaper than building the one with the entire data set; the computational cost of Gaussian process inference scales approximately on the number of simulations used in training cubed. This can overcome through a myriad of choices such as using local approximations (Gramacy and Apley 2015), partitioning the parameter space (Park and Apley 2018), leveraging structured data (Plumlee et al. 2021) or simply brute force computing (Wang et al. 2019). However, with all options less data leads to faster inference for both building and evaluating the emulation model, which is a benefit to conducting time-sensitive calibration. Thus the proposed filter-emulate-calibrate procedure will lead to calibration analysis that requires less computation time than the direct emulate-calibrate procedure.

The rest of this paper is organized as follows. Section 2 provides the summary of the COVID-19 simulation model and explains the natural benefit of filtration in this setting. Section 3 briefly overviews the proposed methodology. Section 4 elaborates on emulation and calibration of COVID-19 simulation model. Section 5 provides some numerical results to show the robustness of the proposed calibration approach.

2 COVID-19 SIMULATION MODEL

In this section, we illustrate how the COVID-19 simulator outcomes diverge from the observed data and how filtration can play a critical role in improving the resulting inference. In the setup investigated in this article, there are four uncertain parameters that govern the sojourn time in the exposed compartment of the SEIR-style model (σ), the recovery times from the asymptomatic (γ_A) and symptomatic (γ_Y) compartments, and the infectiousness of an asymptomatic individual relative to an infectious individual (ω_A). In this study, we focus on a deterministic simulator meaning that for a given $\boldsymbol{\theta} = (\sigma, \omega_A, \gamma_Y, \gamma_A)$, the simulator generates the same output for each replicate. The parameters and their prior distributions are given in Table 1. We consider this setting sufficiently complex to explore the ideas discussed in this article, but one could easily apply this methodology to a stochastic simulator with an increased number of parameters.

Table 1: COVID-19 epidemic model parameters and their prior ranges.

Parameter	Label	Prior
σ	θ_1	Uniform(1, 5)
ω_A	θ_2	Uniform(0.1, 5)
γ_Y	θ_3	Uniform(1, 7)
γ_A	θ_4	Uniform(1, 7)

Our goal is to conduct inference using COVID-19 hospitalization data (specifically hospital census, ICU census, and daily hospital admissions) from February 28 through October 7, 2020 from the Austin, Texas metropolitan area, with a population of about 2.2 million (City of Austin 2020). The simulation model was used to construct the optimal staged alert system in Austin (Yang et al. 2021), and the calibration of this simulation model has been done regularly because the data and the model have been updated frequently with different public health interventions. Figure 1 illustrates simulator outputs and the observed data for hospital census (1st row), ICU census (2nd row), and daily hospital admissions (3rd row). As in Figure 1a, the simulator outputs with a total of 1000 random samples have greater degree of uncertainty in model predictions before filtering. Given the data set, such as the COVID-19 hospitalizations in Austin, a subset of simulator outputs can be filtered out in order to restrict ourselves to a certain parameter space to do inference.

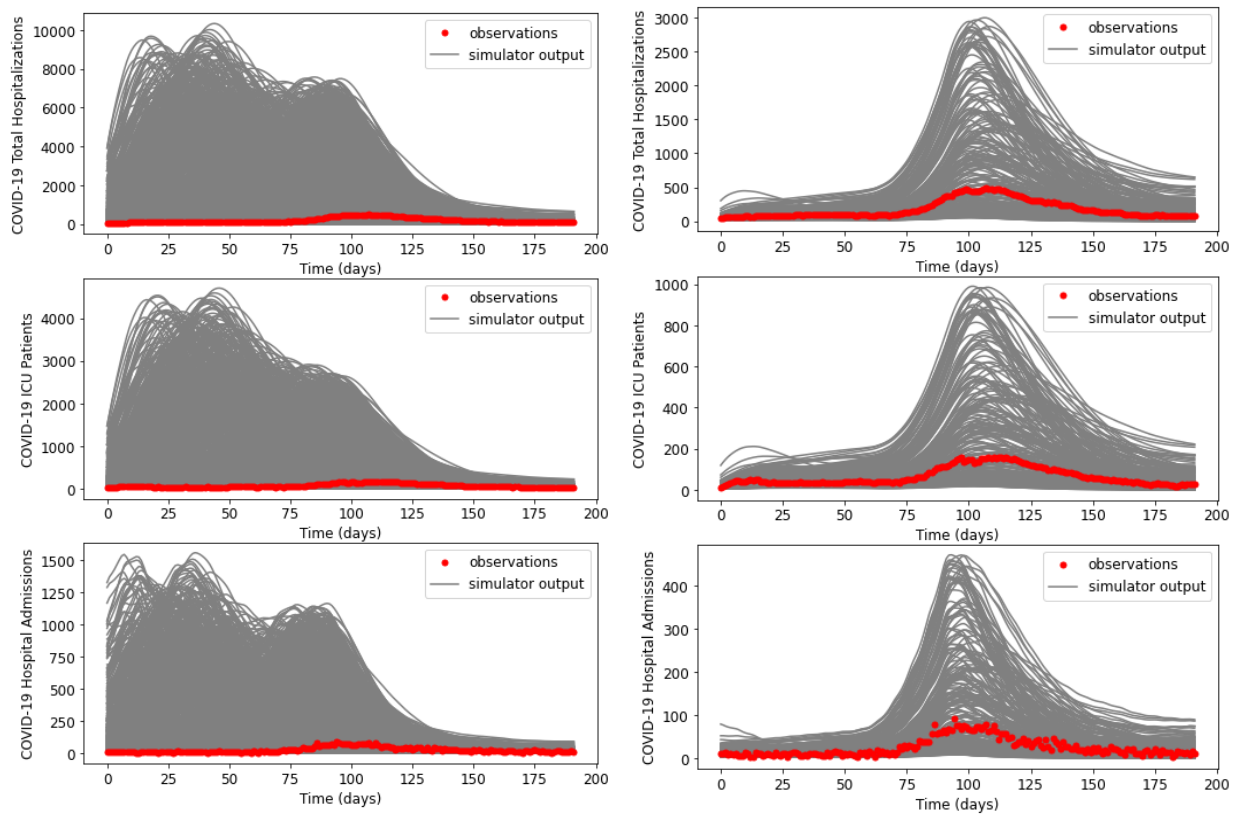
In this example, similar to issues faced by simulation researchers studying models in nascent stages, there are unreasonable results at some parameter combinations. Our suggestion is to filter these unreasonable results out from the simulation results. For this example, the filtration is done by removing the simulator outputs with census hospitalization at day 100 (which corresponds to day when the first peak occurs in hospitalizations) that are either smaller than 50 or larger than 3000. The resulting 206 filtered simulator outputs are illustrated in Figure 1b. The goal of filtering here is to remove parameters with unrealistic simulation outcomes rather than identify parameters with realistic simulator outcomes. For example, the threshold of 3000 is over the Austin area total hospital capacity of 1500, and the values over 3000 do not reflect the realistic case.

3 OVERVIEW OF THE PROPOSED CALIBRATION FRAMEWORK

3.1 Review of Bayesian Calibration

Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ denote the set of calibration parameters that are unknown or uncertain. We assume that there exists a corresponding physical observation from the real system denoted by vector $\mathbf{y} = (y_1, \ldots, y_d)$ and it can be modeled as the best fit of the complex simulation model $\boldsymbol{\eta}(\boldsymbol{\theta})$. Our statistical model is of the form

$$\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\theta}) + \boldsymbol{\varepsilon},$$



(a) Initial simulator outputs.

(b) Filtered simulator outputs.

Figure 1: COVID-19 simulator outputs for Austin, from February 28, 2020 through October 7, 2020. In both plots, the grey curves indicate deterministic simulations, the red points correspond to the reported COVID-19 admissions, hospital census, and ICU census for all Austin area hospitals.

where $\boldsymbol{\varepsilon}$ denotes the residual error and is assumed to come from a multivariate normal distribution with a mean vector of zeros and a covariance matrix (assumed known) $\boldsymbol{\Sigma}$. Importantly, the model is deterministic and the discrepancy between the model output $\boldsymbol{\eta}(\boldsymbol{\theta})$ and the observations $\mathbf{y}$ is independent of $\boldsymbol{\theta}$. There are some ways to account for emulation for stochastic simulations (Ankenman et al. 2010; Plumlee and Tuo 2014; Binois et al. 2018), but this complexity is ignored in this article for brevity.

As with all Bayesian inference, we assume that we are provided a prior distribution of the unknown calibration parameter $\boldsymbol{\theta}$ denoted by $p(\boldsymbol{\theta})$ and the likelihood $p(\mathbf{y}|\boldsymbol{\theta})$. The prior distribution $p(\boldsymbol{\theta})$ is generally based on subject-matter expert opinion. Using Bayes rule, the posterior density has the form

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}),$$

where $\propto$ implies equality up to some constant. For our case, the likelihood is proportional to

$$p(\mathbf{y}|\boldsymbol{\theta}) = |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{ -\frac{1}{2}(\mathbf{y}-\boldsymbol{\eta}(\boldsymbol{\theta}))^{\intercal}\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\eta}(\boldsymbol{\theta}))\right\}. \tag{1}$$

If a simulator is not computationally demanding, then the inference would be done using MCMC on this form. However, if an evaluation of a simulation model is computationally expensive, the emulator replaces the simulation model to predict the simulator output. In that case, the likelihood is updated using the emulator's mean and covariance. Before we discuss the details of the emulator in Section 4, we briefly overview the proposed calibration approach in Section 3.2.

3.2 Proposed Filtering-Based Bayesian Framework

Say that our prior on the parameters is sufficiently broad in the sense that draws from the prior $p(\boldsymbol{\theta})$ can be such that $\boldsymbol{\eta}(\boldsymbol{\theta})$ yield unreasonable results. Pre-processing operations, like those discussed in Section 2, can be used to remove unwanted simulator outputs via some filtration rules. For the rest of the paper, we use the terms "accept" and "reject" to denote the filtered-in and filtered-out simulator outputs, respectively. In that sense, we "accept" a simulator output if it is reasonable, and otherwise we "reject" and exclude because it is unreasonable. Let the acceptability function $r(\boldsymbol{\theta})$ be a binary variable representing acceptability such that

$$r(\boldsymbol{\theta}) = \begin{cases} 1 \text{ if a simulator output is accepted for the corresponding parameter } \boldsymbol{\theta}, \text{ and} \\ 0 \text{ otherwise.} \end{cases}$$

The terms an accepted simulator output and an accepted parameter can be used interchangeably because there is one-to-one correspondence between a parameter and a corresponding simulation output.

The likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ then can be written as

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(\mathbf{y}, r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta}) + p(\mathbf{y}, r(\boldsymbol{\theta}) = 0|\boldsymbol{\theta}).$$

When $r(\boldsymbol{\theta}) = 0$, the simulator output generated with $\boldsymbol{\theta}$ is unreasonable and thus drastically diverges from the observed data $\mathbf{y}$, and thus $p(\mathbf{y}, r(\boldsymbol{\theta}) = 0|\boldsymbol{\theta}) \approx 0$. Given $p(\mathbf{y}, r(\boldsymbol{\theta}) = 0|\boldsymbol{\theta}) \approx 0$, we can approximate $p(\mathbf{y}|\boldsymbol{\theta})$ as

$$p(\mathbf{y}|\boldsymbol{\theta}) \approx p(\mathbf{y}, r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta}). \tag{2}$$

Using conditional probability, (2) can be written equivalently as

$$p(\mathbf{y}|\boldsymbol{\theta}) \approx p(\mathbf{y}, r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta}) = p(\mathbf{y}|\boldsymbol{\theta}, r(\boldsymbol{\theta}) = 1)p(r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta}). \tag{3}$$

It helps to contrast (3) with an ad hoc approach. In practice, filtration often happens naturally during analysis where one informally explores the parameter space to find decent regions of the parameter space and then emulates where the simulator outputs are deemed reasonable. In this case, given the filtered simulator outputs (e.g., the simulator outputs with $r(\boldsymbol{\theta}) = 1$, and the ones with $r(\boldsymbol{\theta}) = 0$ are filtered out) we would compute the likelihood via

$$p(\mathbf{y}|\boldsymbol{\theta}) \approx p(\mathbf{y}|\boldsymbol{\theta}, r(\boldsymbol{\theta}) = 1). \tag{4}$$

On an intuitive level, this procedure makes sense as the emulator only needs to be accurate where there are reasonable outputs. However, if one uses the emulation built on filtered data without accounting for the filtration, the inference can be wrong. The calibration posterior will be much larger than desired in a way that keeps mass in areas where there is little chance of getting acceptable simulation outcomes. This is incorrect because the emulation built on filtered data is the prediction conditional on the model output being acceptable.

One must use (3) instead of (4) in order for filtration and emulation to produce reasonable inference. This amounts to a simple fix: multiply the likelihood from filtered responses by an adjustment factor $p(r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta})$ in (3) that reflects the probability of $\boldsymbol{\theta}$ being accepted. The development of this is described in Section 4.4. This way, regions with little likelihood of acceptability will have reduced posterior mass even if the likelihood conditional on it being accepted is large. In this paper, we integrate this adjustment factor into the calibration framework as a multiplicative of the likelihood $p(\mathbf{y}|\boldsymbol{\theta}, r(\boldsymbol{\theta}) = 1)$ conditioning on the corresponding simulator outcome is being accepted as it will be described in Section 4.5.

4 EMULATION AND CALIBRATION OF COVID-19 SIMULATION

4.1 Emulation Model for the Simulation Output

Calibration often involves running a simulator many times to draw samples from the posterior of $\boldsymbol{\theta}$. If the simulator runs fast, one can compute the likelihood as in (1) using the simulator response. However, in

many cases a direct use of simulation model $\boldsymbol{\eta}(\boldsymbol{\theta})$ for MCMC is not practical because it is computationally inefficient to run the simulation even for a single parameter vector $\boldsymbol{\theta}$. An emulator is commonly used as a surrogate to replace the simulator in these cases to reduce the computational cost. We treat $\boldsymbol{\eta}(.)$ as an unknown function and use the responses from n initial parameter settings as a training set to emulate $\boldsymbol{\eta}(.)$ leveraging a Gaussian process (GP) model. These initial parameters are randomly generated from their prior distribution.

For the rest of this section, $\boldsymbol{\theta}$ denotes the unknown best parameter setting, and $\mathbf{t}$ represents any parameter setting. To construct an emulator, we model the simulation output using the basis representation of Higdon et al. (2008) where

$$\boldsymbol{\eta}(\mathbf{t}) = \sum_{i=1}^{q} \mathbf{k}_i w_i(\mathbf{t}) + \boldsymbol{e} \tag{5}$$

where $\{\mathbf{k}_1, \ldots, \mathbf{k}_q\}$ is a collection of orthogonal d-dimensional basis vectors, $w_i(\mathbf{t})$ are weights modeled with independent, univariate GPs and $\boldsymbol{e}$ is an d-dimensional term that contains any residual between the model and the span of $\{\mathbf{k}_1, \ldots, \mathbf{k}_q\}$. The COVID-19 simulator requires $p = 4$ dimensional parameter vector $\mathbf{t} = (t_1, t_2, t_3, t_4)$ (see Table 1), and returns $d = 114$ dimensional output.

In (5), the weight $w_i(\mathbf{t})$ for $i = 1, \ldots, q$ is a function mapping the p-dimensional input $\mathbf{t}$ to a scalar. These q functions are modeled as independent GPs on the region where $r(.) = 1$ with mean and covariance function $\gamma_i(.)$ and $C_i(.,.)$. Explained differently, if Θ is our parameter space, then we are presuming that

$$w_i(.) \sim \mathrm{GP}(\gamma_i(.), C_i(.,.)) \text{ on } \Theta_A = \{\mathbf{t} | r(\mathbf{t}) = 1, \mathbf{t} \in \Theta\}. \tag{6}$$

We presume that $w_i(.)$ is independent of all locations where $r(\boldsymbol{\theta}) = 0$. Following typical emulation approaches (Gramacy 2020), we take the covariance as some constant times a correlation function plus some nugget (Gramacy and Lee 2012),

$$C_i(\mathbf{t}, \mathbf{t}') = \tau_i^2 \left[R(\mathbf{t}, \mathbf{t}'; \boldsymbol{\rho}_i) + \nu_i \delta_{\mathbf{t},\mathbf{t}'} \right], \tag{7}$$

where $\delta_{\mathbf{t},\mathbf{t}'} = 1$ if $\mathbf{t} = \mathbf{t}'$ and 0 otherwise. For the correlation function R, we used the separable version of the Matérn correlation function with smoothness parameter 1.5 (Handcock and Stein 1993) which depends on a p-dimensional vector of length-scale parameters $\boldsymbol{\rho}_i$. In (7), ν_i and $\boldsymbol{\rho}_i$ are the hyper-parameters that can be estimated from the training data by numerically maximizing the likelihood function as described in the next subsection. The key to our inference approach is that this setup, namely (6), implies we are only interested in the parameters that were not filtered out.

To find $\mathbf{k}_1, \ldots, \mathbf{k}_q$, we use the following process. Using $n = 206$ filtered simulator runs corresponding to parameter settings $\mathbf{t}_1^*, \ldots, \mathbf{t}_n^*$, and obtain d-dimensional simulator outputs $\boldsymbol{\eta}(\mathbf{t}_1^*), \ldots, \boldsymbol{\eta}(\mathbf{t}_n^*)$, and then construct $d \times n$ data matrix $\Xi = [\boldsymbol{\eta}(\mathbf{t}_1^*), \ldots, \boldsymbol{\eta}(\mathbf{t}_n^*)]$. After mean-centering and scaling each of the rows of Ξ, we obtain the q orthogonal basis vectors $\mathbf{k}_1, \ldots, \mathbf{k}_q$ via principal components to project the original simulator outputs onto a lower-dimensional space spanned by an orthogonal basis. In this case study, less than 10 PCs are enough to explain 99% of the variation in the simulator output Ξ. Using n parameter settings, let the n-vector $\mathbf{w}_i$ be $\mathbf{w}_i = (w_i(\mathbf{t}_1^*), \ldots, w_i(\mathbf{t}_n^*))$ for $i = 1, \ldots, q$, and let $n \times n$ matrix $\mathbf{C}_i$ be the covariance matrix resulting from applying (7) to each pair of the parameter settings $\mathbf{t}_1^*, \ldots, \mathbf{t}_n^*$. Then, the nq-vector $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_q)$ has a prior distribution

$$\begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_q \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \boldsymbol{\gamma}_1 \\ \vdots \\ \boldsymbol{\gamma}_q \end{pmatrix}, \begin{pmatrix} \mathbf{C}_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \mathbf{C}_q \end{pmatrix} \right).$$

4.2 Estimation of the Hyper-parameters in the Gaussian Process Model

Following common procedures and to eliminate a very expensive numeric integration problem, we will set all hyper-parameters equal to their maximum likelihood data given the (accepted) simulation responses.

The GP prior on $\mathbf{w}_i$, that is, $\mathbf{w}_i \sim \mathrm{N}(\boldsymbol{\gamma}_i, \mathbf{C}_i)$ as stated above, implies that the likelihood is in the form of

$$(2\pi\tau_i^2)^{-n/2}|\mathbf{C}_i|^{-1/2}\exp\left\{-\frac{1}{2\tau_i^2}(\mathbf{w}_i-\boldsymbol{\gamma}_i)^\intercal\mathbf{C}_i^{-1}(\mathbf{w}_i-\boldsymbol{\gamma}_i)\right\}. \tag{8}$$

In order to simplify the likelihood further, we assume a constant mean for each GP such that $\boldsymbol{\gamma}_i = \gamma_i\mathbf{1}_n$. Taking the log of (8) and plugging $\boldsymbol{\gamma}_i = \gamma_i\mathbf{1}_n$ yields the log-likelihood

$$l(\tau_i^2, \boldsymbol{\gamma}_i, \mathbf{C}_i, \nu_i, \boldsymbol{\rho}_i) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\tau_i^2) - \frac{1}{2}\log(|\mathbf{C}_i|) - \frac{1}{2\tau_i^2}(\mathbf{w}_i-\gamma_i\mathbf{1}_n)^\intercal\mathbf{C}_i^{-1}(\mathbf{w}_i-\gamma_i\mathbf{1}_n). \tag{9}$$

Maximizing the log-likelihood with respsect to τ_i^2 and γ_i gives us

$$\hat{\gamma}_i = \frac{\mathbf{1}_n^\intercal\mathbf{C}_i^{-1}\mathbf{w}_i}{\mathbf{1}_n^\intercal\mathbf{C}_i^{-1}\mathbf{1}_n} \text{ and } \hat{\tau}_i^2 \quad = \frac{1}{n}(\mathbf{w}_i-\hat{\gamma}_i\mathbf{1}_n)^\intercal\mathbf{C}_i^{-1}(\mathbf{w}_i-\hat{\gamma}_i\mathbf{1}_n). \tag{10}$$

The hyperparameters ν_i and $\boldsymbol{\rho}_i$ need to be estimated as well. If we plug the estimates $\hat{\gamma}_i$ and $\hat{\tau}_i^2$ in (10) back into our log-likelihood in (9), we can ignore an additive constant and get a revised log-likelihood of

$$l^\star(\nu_i, \boldsymbol{\rho}_i) = -\frac{n}{2}\log(\hat{\tau}_i^2) - \frac{1}{2}\log(|\mathbf{C}_i|).$$

Maximizing $l^\star(\nu_i, \boldsymbol{\rho}_i)$ requires numerical methods. In this study, all hyper-parameters are optimised until convergence using `scipy`'s implementation of the L-BFGS-B algorithm (Nocedal and Wright 2006).

4.3 Emulator-based Predictions

We require the ability to infer on $\boldsymbol{\eta}(.)$ for any parameter input conditional on it being accepted. In order to generalize predictions at any parameter settings, we use conditional normal theory, and find that

$$w_i(\boldsymbol{\theta})|\mathbf{w}_i, r(\boldsymbol{\theta}) = 1 \sim \mathrm{N}(\mu_i(\boldsymbol{\theta}), \sigma_i^2(\boldsymbol{\theta},\boldsymbol{\theta}))$$

with mean $\mu_i(\boldsymbol{\theta}) = \hat{\gamma}_i + \mathbf{c}_i(\boldsymbol{\theta})^\intercal\mathbf{C}_i^{-1}(\mathbf{w}_i - \hat{\gamma}_i\mathbf{1}_n)$ and variance $\sigma_i^2(\boldsymbol{\theta},\boldsymbol{\theta}) = C_i(\boldsymbol{\theta},\boldsymbol{\theta}) - \mathbf{c}_i(\boldsymbol{\theta})^\intercal\mathbf{C}_i^{-1}\mathbf{c}_i(\boldsymbol{\theta})$. Here, $\mathbf{c}_i(\boldsymbol{\theta})$ is an n-dimensional vector obtained by applying (7) to n experimental settings $\{\mathbf{t}_1^*, \ldots, \mathbf{t}_n^*\}$ crossing with any setting $\boldsymbol{\theta}$.

Let $d \times q$ matrix $\mathbf{K}$ be $\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_q]$, and let q-length vector $\mathbf{w}(\boldsymbol{\theta}) = (w_1(\boldsymbol{\theta}), \ldots, w_q(\boldsymbol{\theta}))$. Define $\boldsymbol{\mu}(\boldsymbol{\theta}) = (\mu_1(\boldsymbol{\theta}), \ldots, \mu_q(\boldsymbol{\theta}))$ and $q \times q$ matrix $\mathbf{S}(\boldsymbol{\theta})$ with diagonal elements $\sigma_i^2(\boldsymbol{\theta},\boldsymbol{\theta})$ for $i = 1, \ldots, q$ and off-diagonal elements 0. Then, because one can generate the realizations from the process $\boldsymbol{\eta}(\boldsymbol{\theta})$ at $\boldsymbol{\theta}$ via $\boldsymbol{\eta}(\boldsymbol{\theta}) = \sum_{i=1}^q \mathbf{k}_i w_i(\boldsymbol{\theta})$, we can obtain

$$\boldsymbol{\eta}(\boldsymbol{\theta}) \sim \mathrm{N}(\mathbf{K}\boldsymbol{\mu}(\boldsymbol{\theta}), \mathbf{K}\mathbf{S}(\boldsymbol{\theta})\mathbf{K}^\intercal). \tag{11}$$

Here, $\mathbf{K}\boldsymbol{\mu}(\boldsymbol{\theta})$ and $\mathbf{K}\mathbf{S}(\boldsymbol{\theta})\mathbf{K}^\intercal$ are the emulator predictive mean and covariance, respectively.

4.4 Probability of Acceptance

In (3), in addition to $p(\mathbf{y}|\boldsymbol{\theta}, r(\boldsymbol{\theta}) = 1)$, the probability of accepting the simulator output for a given input parameter $\boldsymbol{\theta}$, that is, $p(r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta})$ is needed, and we use a statistical-learning method to estimate $p(r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta})$ as follows. We construct a training data set where each observation of data corresponds to one simulation run, and the training set consists of 400 randomly selected balanced rows with 200 accepted and 200 rejected simulator runs. We fit a binary classification model to predict whether the simulator output is accepted (the response) based on a given input parameter $\boldsymbol{\theta}$ (the predictors). The response is defined as a binary random variable that takes values zero and one, and it is one if the simulator output corresponding

to $\boldsymbol{\theta}$ is accepted, otherwise zero, according to user chosen filtration rules like those described in Section 2. In order to estimate $p(r(\boldsymbol{\theta})=1|\boldsymbol{\theta})$, we consider the following logistic regression model

$$\log\left[\frac{p(r(\boldsymbol{\theta})=1|\boldsymbol{\theta})}{1-p(r(\boldsymbol{\theta})=1|\boldsymbol{\theta})}\right]=\beta_0+\beta_1\theta_1+\cdots+\beta_4\theta_4+\beta_5\theta_1^2+\cdots+\beta_8\theta_4^2+\beta_9\theta_1\theta_2+\cdots+\beta_{14}\theta_3\theta_4, \quad (12)$$

where each β_i is a regression coefficient for $i=0,\ldots,14$. We estimate these β_is via `sklearn` (Pedregosa et al. 2011), and include the quadratic and interaction terms as in (12) since including them increases the classification accuracy (i.e., the percentage of correct predictions out of total number of samples) on the test data about 10% (increased from 74% to 85%).

4.5 Calibration of COVID-19 Simulation Model with the Proposed Approach

In this study, we use the emulator described above to replace the computationally expensive COVID-19 simulation model. In such a case, when computing the likelihood $p(\mathbf{y}|\boldsymbol{\theta})$, the simulator output $\boldsymbol{\eta}(\boldsymbol{\theta})$ is replaced by the emulator mean, and the prediction uncertainty from the emulator is integrated within the likelihood. We assume that the emulator is conditionally a GP given $r(\boldsymbol{\theta})=1$ as in (6), and it is fitted using n filtered simulation runs as explained above. At any input setting, we obtain the emulator mean and covariance as $\mathbf{K}\boldsymbol{\mu}(.)$ and $\mathbf{KS}(.)\mathbf{K}^\intercal$, respectively, as given in (11).

Using the idea introduced in Section 3.2, we compute the posterior of $\boldsymbol{\theta}$ via

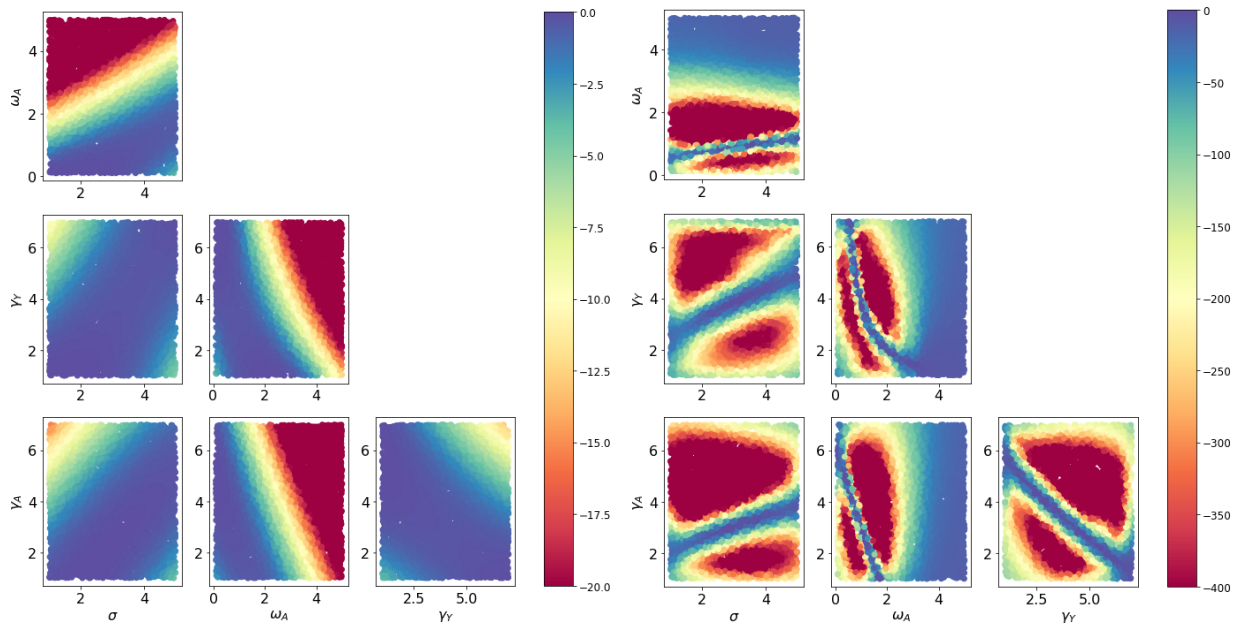
$$p(\boldsymbol{\theta}|\mathbf{y})\propto p(\mathbf{y}|\boldsymbol{\theta},r(\boldsymbol{\theta})=1)p(r(\boldsymbol{\theta})=1|\boldsymbol{\theta})p(\boldsymbol{\theta}). \quad (13)$$

Integrating emulation mean and variance to the likelihood, one can obtain the conditional likelihood $p(\mathbf{y}|\boldsymbol{\theta},r(\boldsymbol{\theta})=1)$ in (13) as

$$|\boldsymbol{\Sigma}+\mathbf{KS}(\boldsymbol{\theta})\mathbf{K}^\intercal|^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{y}-\mathbf{K}\boldsymbol{\mu}(\boldsymbol{\theta}))^\intercal(\boldsymbol{\Sigma}+\mathbf{KS}(\boldsymbol{\theta})\mathbf{K}^\intercal)^{-1}(\mathbf{y}-\mathbf{K}\boldsymbol{\mu}(\boldsymbol{\theta}))\right\}. \quad (14)$$

5 RESULTS

In this section, we present our results and provide insights into why incorporating the probability of acceptability into the calibration framework helps to avoid unstable and incorrect inference. First, we explain the role of the acceptance probability in the proposed calibration approach. As described above, we use four uncertain parameters, that is, $\boldsymbol{\theta}=(\sigma,\omega_A,\gamma_Y,\gamma_A)$, to calibrate, and Figure 2 illustrates how the log of the acceptance probability $p(r(\boldsymbol{\theta})=1|\boldsymbol{\theta})$ (in Figure 2a) and the log of the likelihood $p(\mathbf{y}|\boldsymbol{\theta},r(\boldsymbol{\theta})=1)$ conditioning on the parameters being accepted (in Figure 2b) vary with changing values of the pair of parameters. In the 3×3 array of panels in Figures 2a–2b, columns correspond to the parameters σ,ω_A,γ_Y and rows correspond to the parameters $\omega_A,\gamma_Y,\gamma_A$, respectively. Within each panel the corresponding values are visualized with changing values of the pair of parameters represented by the x- and y- axes, and the remaining two parameters are set to the median of the 1000 samples from the posterior distribution of $\boldsymbol{\theta}$ determined from MCMC.



(a) Log-acceptance probability, $\log p(r(\boldsymbol{\theta}) = 1|\boldsymbol{\theta})$.

(b) Log-likelihood, $\log p(\mathbf{y}|r(\boldsymbol{\theta}) = 1, \boldsymbol{\theta})$.

Figure 2: Illustration of the log-acceptance probability and the log-likelihood with changing values of all paired combinations of parameters.

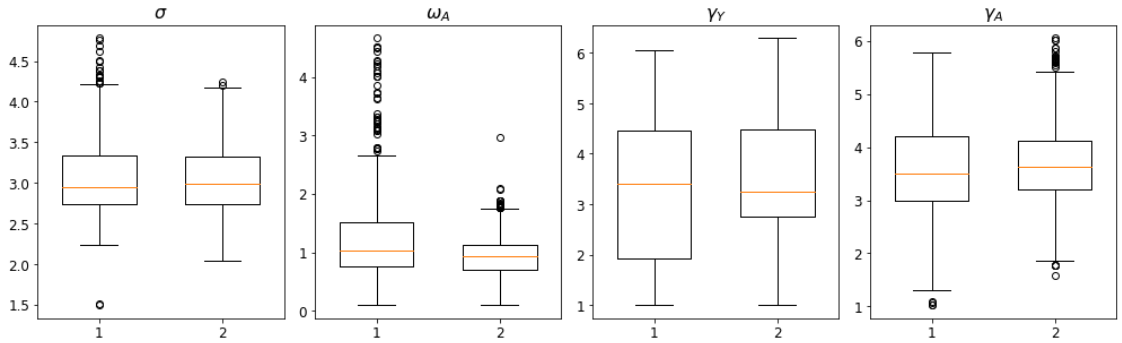
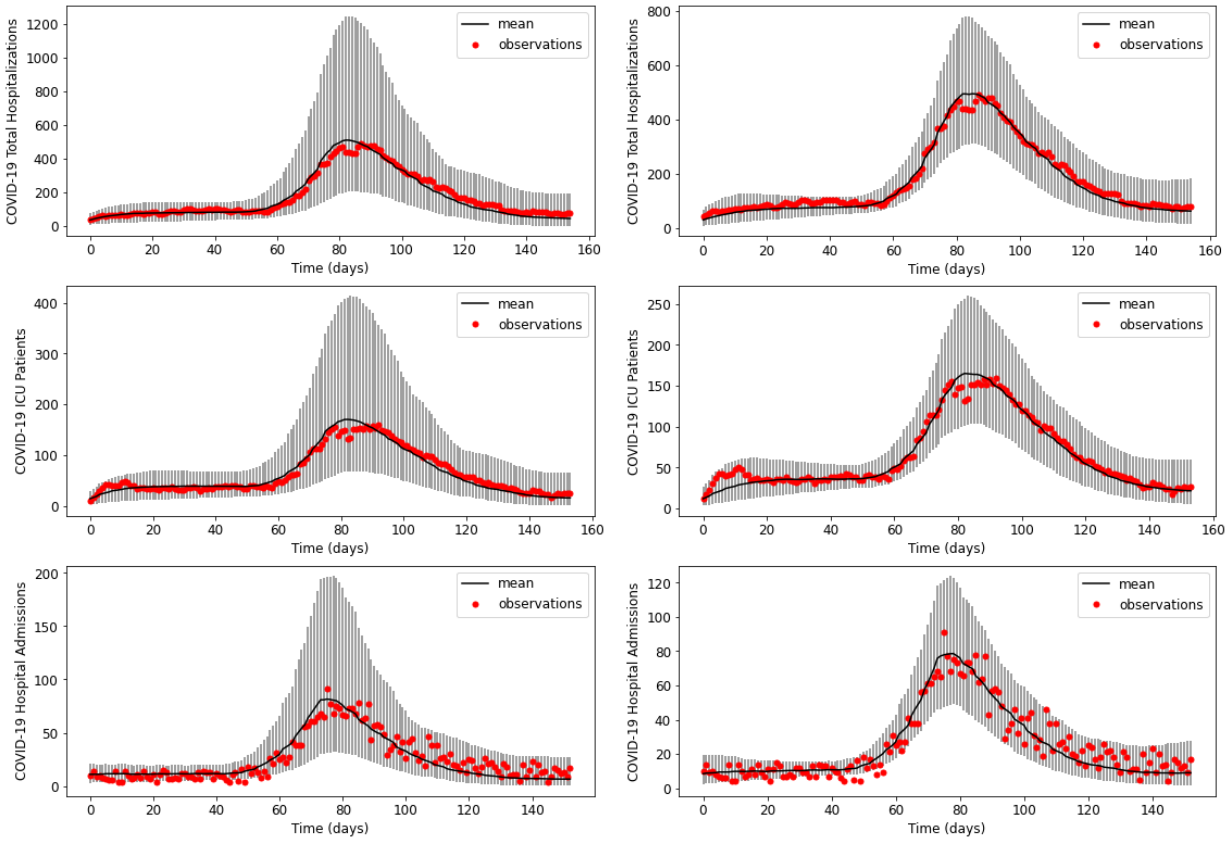


Figure 3: Distribution of the calibrated parameters σ, ω_A, γ_Y, γ_A sampled from the posterior distribution. The results are obtained through calibration using emulation of filtered simulation results without acceptance probability (1) and with acceptance probability (2).



(a) Calibration without acceptance probability.

(b) Calibration with acceptance probability.

Figure 4: Emulator mean at posterior parameters obtained at test points through calibration using emulation of filtered simulation results without acceptance probability (left) and with acceptance probability (right). In both plots, the grey lines indicate the error bars corresponding to the 95% interval, the red points correspond to the reported COVID-19 hospital census, ICU census and hospital admissions for all Austin area hospitals.

Figure 2a illustrates the log of the probability that the simulator output is being accepted for a given parameter $\boldsymbol{\theta}$. In Figure 2b, the log-likelihood values are illustrated for the same pair of parameters. We can now revisit the pitfall inference discussed in Section 3. If one applies the aforementioned filter-emulate-calibrate procedure without considering the acceptance probability, then the log-likelihood would look like Figure 2b. Obviously, this leads to large posterior density for the parameters that are likely to return both realistic and unrealistic simulator outputs. The parameters lying in the upper blue region in the initial panel in Figure 2b for instance tend to be the rejected ones in the upper red region of Figure 2a. If one does not consider the fact that the inference is conditional on the simulator output is being acceptable and ignores the acceptance probability, then the inference will be wrong. On the other hand, when the acceptance probability is incorporated into the calibration framework as in the proposed approach, the parameters lying in the regions with smaller probability of acceptance in Figure 2a have reduced adjusted likelihood, and thus the reduced posterior density. This illustrates exactly why including the probability of acceptance is critical, because without it, unrealistic parameters would have large mass in the posterior.

Recall that we obtained the likelihood in (14) using the emulator built on the filtered simulator outputs. As mentioned above, the advantages of emulating the simulator outputs over a filtered space are the decreased computational expense and the improved prediction quality. For example, in our case study if

we built an emulator without filtering out any simulator outputs (e.g., using all simulator outputs illustrated in Figure 1a), then the number of observations in the training set would be 1000. In such a case, the computational time of training and prediction procedure would increase compared to that of the emulator fitted with 206 observations. In terms of the emulation prediction quality, as an illustration, in addition to the emulator fitted with the filtered simulation outputs, we built an emulator with all simulator outputs, and evaluated the prediction quality on a test set via the root-mean-square error (RMSE) value. We obtained the RMSE of 3.66 (with 17 PCs) using the emulator fitted with 1000 outputs, whereas the RMSE is 1.27 (with 9 PCs) using the emulator built with 206 filtered simulator outputs. This demonstrates that by building an emulator with filtered data we can obtain higher prediction accuracy with less number of PCs because filtering eliminates the unstable outcomes. Once we fit the emulator with filtered outputs and incorporate the acceptance probability into the posterior, we use the parallel-tempering ensemble MCMC (Liu 2008, Chapter 10) to sample from the posterior. This posterior is very difficult for most MCMC approaches because of multi-modality. This is especially true for a method that ignores the acceptance probability term (see the log-likelihood illustrated in Figure 2b), further motivating the need for including this novel acceptance probability term.

We compare the proposed approach with the filter-emulate-calibrate procedure without the acceptance probability in our benchmark study. The first calibration approach requires the emulator fitted with filtered simulation outputs, and returns the posterior draws without using the acceptance probability in computation of the posterior. The second one is the proposed approach, and the posterior is obtained via (13). Figure 3 illustrates the variability in the median and the ranges of the parameters with two different approaches. As the ranges of the four parameters show, the incorporation of a statistical-learning method in the form of an acceptance probability into the calibration formulation indicate a reduction of uncertainty levels in the parameters, especially for the second parameter ω_A.

Once we obtain the posterior draws by using two different approaches, we predict the COVID-19 hospital census, ICU census and hospital admissions at unseen time points (we downsampled our 576-dimensional response vector to 114 dimensions, leaving the remaining for testing). Figure 4 illustrates the observed data, the prediction mean (black line), and the prediction error at each time point (grey lines) without using an acceptance probability and with using an acceptance probability. Overall, as the calibration results indicate our approach greatly reduces the uncertainty in the model predictions. This comparison has helped us identify the trade-off in not using acceptance probability when the filtered data is used through calibration in terms of reducing the uncertainty in the parameters and the resulting predictions.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support from National Science Foundation grants OAC 2004601 and DMS 1953111.

REFERENCES

Ankenman, B., B. L. Nelson, and J. Staum. 2010. "Stochastic Kriging for Simulation Metamodeling". *Operations Research* 58(2):371–382.

Baker, E., P. Barbillon, A. Fadikar, R. B. Gramacy, R. Herbei, D. Higdon, J. Huang, L. R. Johnson, P. Ma, A. Mondal, B. Pires, J. Sack, and V. Sokolov. 2020. "Analyzing Stochastic Computer Models: A Review with Opportunities". *arXiv preprint arXiv:2002.01321*.

Beaumont, M. A., W. Zhang, and D. J. Balding. 2002. "Approximate Bayesian computation in population genetics". *Genetics* 162(4):2025–2035.

Binois, M., R. B. Gramacy, and M. Ludkovski. 2018. "Practical heteroscedastic gaussian process modeling for large simulation experiments". *Journal of Computational and Graphical Statistics* 27(4):808–821.

City of Austin 2020, November. "Key Indicators For Staging".

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2013. *Bayesian data analysis*. 3 ed. CRC press.

Gramacy, R. B. 2020. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. New York, NY: CRC Press ; Taylor & Francis Group.

Gramacy, R. B., and D. W. Apley. 2015. "Local Gaussian process approximation for large computer experiments". *Journal of Computational and Graphical Statistics* 24(2):561–578.

Gramacy, R. B., and H. K. Lee. 2012. "Cases for the nugget in modeling computer experiments". *Statistics and Computing* 22(3):713–722.

Handcock, M. S., and M. L. Stein. 1993. "A Bayesian analysis of kriging". *Technometrics* 35(4):403–410.

Higdon, D., J. Gattiker, B. Williams, and M. Rightley. 2008. "Computer Model Calibration Using High-Dimensional Output". *Journal of the American Statistical Association* 103(482):570–583.

Liu, J. S. 2008. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

Nocedal, J., and S. J. Wright. 2006. *Numerical Optimization*. second ed. New York, NY, USA: Springer.

Park, C., and D. Apley. 2018. "Patchwork kriging for large-scale gaussian process regression". *The Journal of Machine Learning Research* 19(1):269–311.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research* 12:2825–2830.

Plumlee, M. 2017. "Bayesian Calibration of Inexact Computer Models". *Journal of the American Statistical Association* 112(519):1274–1285.

Plumlee, M. 2019. "Computer model calibration with confidence and consistency". *Journal of the Royal Statistical Society: B* 81(3):519–545.

Plumlee, M., C. Erickson, B. Ankenman, and E. Lawrence. 2021. "Composite grid designs for adaptive computer experiments with fast inference". *Biometrika, to appear*.

Plumlee, M., and R. Tuo. 2014. "Building accurate emulators for stochastic simulations via quantile kriging". *Technometrics* 56(4):466–473.

Rutter, C. M., J. Ozik, M. DeYoreo, N. Collier et al. 2019. "Microsimulation model calibration using incremental mixture approximate Bayesian computation". *Annals of Applied Statistics* 13(4):2189–2212.

Santner, T. J., B. J. Williams, and W. I. Notz. 2018. *The Design and Analysis of Computer Experiments*. 2nd ed. 2018 ed. Springer Series in Statistics. New York, NY: Springer New York : Imprint: Springer.

Wang, B., Q. Zhang, and W. Xie. 2019. "Bayesian sequential data collection for stochastic simulation calibration". *European Journal of Operational Research* 277(1):300–316.

Wang, K., G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. 2019. "Exact Gaussian Processes on a Million Data Points". In *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Volume 32: Curran Associates, Inc.

Wang, X., R. F. Pasco, Z. Du, M. Petty, S. J. Fox, A. P. Galvani, M. Pignone, S. C. Johnston, and L. A. Meyers. 2020, October. "Impact of Social Distancing Measures on Coronavirus Disease Healthcare Demand, Central Texas, USA". *Emerging Infectious Diseases* 26(10):2361–2369.

Wong, R. K., C. B. Storlie, and T. C. Lee. 2017. "A frequentist approach to computer model calibration". *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 79(2):635–648.

Yang, H., O. Sürer, D. Duque, D. P. Morton, B. Singh, S. J. Fox, R. Pasco, K. Pierce, P. Rathouz, V. Valencia, Z. Du, M. Pignone, M. E. Escott, S. I. Adler, S. C. Johnston, and L. A. Meyers. 2021. "Design of COVID-19 Staged Alert Systems to Ensure Healthcare Capacity with Minimal Closures". *Nature Communications* 12(1):3767.

Yuan, J., and S. H. Ng. 2020. "An Integrated Method for Simultaneous Calibration and Parameter Selection in Computer Models". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 30(1):1–23.

Yuan, J., S. H. Ng, and K. L. Tsui. 2012. "Calibration of stochastic computer models using stochastic approximation methods". *IEEE Transactions on Automation Science and Engineering* 10(1):171–186.

AUTHOR BIOGRAPHIES

ÖZGE SÜRER is a postdoctoral research fellow at the Northwestern-Argonne Institute of Science and Engineering. Her research interests lie in the intersection of statistical learning and optimization. Her e-mail address is ozgesurer2019@u.northwestern.edu. Her website is https://ozgesurer.github.io/OS/.

MATTHEW PLUMLEE is an Assistant Professor in the Department of Industrial Engineering and Management Sciences at Northwestern University. He primarily researches uncertainty quantification methods for computational models of systems. His e-mail address is mplumlee@northwestern.edu. His website is http://users.iems.northwestern.edu/ mplumlee/.