A unified framework for computer model calibration and engineering design

Carl Ehrett*
School of Mathematical and Statistical Sciences, Clemson University,
D. Andrew Brown
School of Mathematical and Statistical Sciences, Clemson University,
Christopher Kitchens
Department of Chemical and Biomolecular Engineering, Clemson University,
Sez Atamturktur
Department of Architectural Engineering, Pennsylvania State University,
Leslie Xu
Department of Architectural Engineering, Pennsylvania State University, and
Roland Platz
TODO AFFILIATION?

Abstract

The calibration of computer models and the use of those computer models for design are two activities that are traditionally carried out separately. This paper generalizes existing Bayesian inverse analysis approaches for computer model calibration to present a methodology that combines calibration and design under a unified Bayesian framework. This unified framework provides a computationally efficient means for undertaking both tasks in a way that quantifies all relevant sources of uncertainty. Specifically, compared with the traditional approach of undertaking design using parameter estimates made during previously completed model calibration, this generalized framework inherently includes uncertainty from the calibration process in the design procedure. In addition, we demonstrate how, when adaptive sampling of the phenomenon of interest is possible, the proposed framework is able to select new sampling locations using both the observations made so far of the real phenomenon and the information contained in the computer model. This is especially useful when the model is misspecified in failing to reflect that the parameter to be calibrated is functionally dependent upon the design inputs that are to be optimized.

*The authors gratefully acknowledge grant CMMI-1934438 from the National Science Foundation (NSF). CE was supported by fellowships through Department of Education GAANN grant P200A150310 and NSF NRT grant 1633608. DAB is also supported by NSF grants EEC-1744497 and OIA-1826715.

Keywords: Gaussian processes, optimization, uncertainty quantification, computer model calibration, engineering design

1 Introduction

This paper connects two distinct areas of research concerning computer models of real phenomena. One area is that of computer model calibration, where the goal is to find a posterior distribution on unknown, or imperfectly known, parameters by calibrating a computer model using real-world observations of the modeled phenomenon. The second area is that of enlisting a computer model for design, using the model to find settings for controllable system inputs such that the resulting system output is optimized with respect to some design goal. These two problems are structurally similar, both involving finding estimates or distributions of model inputs to achieve some desired effect on model outputs. In the case of calibration, the desired effect is that the model outputs approximate reality, and in the case of design, the desired effect is that the model outputs approximate the optimal achievable outputs. Since calibration and design are typically carried out separately, existing design techniques operate under the assumption that the model is an accurate approximation of the real system of interest. In practice, models used for design typically are known or suspected to be biased representations of the phenomenon of interest, and often have inputs that require calibration. The goal of the work described here is to provide a unified framework for calibration and design. We refer to this new approach as DCTO, for dual calibration to target outcomes. In addition to avoiding the idealization that the model used for design is unbiased, DCTO allows one to focus calibration efforts on regions of interest, prioritizing them over other areas of the model range. For example, one may be more interested in calibrating the model to be accurate in the optimal region of some design variable $\boldsymbol{\theta}_d$ than elsewhere. Having a combined framework for calibration and

design is especially of interest when those two activities are non-trivially intertwined, as in the case when the value of the calibration parameters are functionally dependent upon the design settings.

Bayesian methods for computer model calibration are developed by Kennedy and O'Hagan (2001). Since their seminal paper, the methodology has seen numerous extensions and refinements (Higdon et al. 2004; Williams et al. 2006; Bayarri et al. 2007a; Bayarri et al. 2007b; Paulo et al. 2012; Brynjarsdóttir and O'Hagan 2014). Henceforth, we refer to this approach to calibration as KOH. Common to KOH approaches is the Bayesian framework in which one places a prior on the calibration parameters $\boldsymbol{\theta}_c$, often pairing it with a Gaussian process (GP) meta-model of the computer model of interest and a GP prior on the model discrepancy $\delta()$, and using the available observations y_r of the real system to find a posterior distribution $\boldsymbol{\theta}_c, \delta()|y_r$. Such an approach is notable for providing not merely a point estimate of the calibration parameter, but for providing a full posterior distribution quantifying remaining uncertainty about $\boldsymbol{\theta}_c$ and about $\delta()$.

Herein, we leverage the KOH framework to find a posterior distribution, not only on unknown model parameters, but also on controllable design settings. We achieve this via an approach called counterfactual Bayes. In traditional model calibration, one uses Bayes' rule to discover a posterior distribution on calibration parameters using real observations, so that the observations are the source of the Bayesian learning. In a design case, there are no relevant observations. One wants to find design settings that induce the system to behave optimally, but one typically has not observed the system doing so, and therefore there seems to be no relevant source of Bayesian learning that could drive the use of Bayes' rule

to discover a posterior distribution on optimal design settings. The idea of counterfactual Bayes is to identify artificial observations, or target outcomes, y_t such that the resulting likelihood is highest in the optimal design region — i.e., target outcomes y_t such that their occurrence is strong evidence that the design settings are optimal. Hence, in addition to calibrating the unknown model parameters against experimental observations, one uses the KOH framework to also find a posterior distribution on design settings given the target outcomes — and given the nature of y_t, this is *de facto* a distribution on optimal design settings for the system. The result retains the benefits of the Bayesian model calibration tools on which it is based, namely the quantification of remaining uncertainty regarding the optimal design settings. And like KOH, DCTO is especially well-suited to problems that rely on black-box functions.

We may divide optimization approaches in such cases broadly into three camps (Regis and Shoemaker, 2004). Gradient-based approaches (Nocedal and Wright, 2006) are of limited utility when dealing with black-box functions, where we cannot evaluate the objective function's derivative. Approximation of the derivative requires additional function evaluations, rapidly inflating the computational cost when each evaluation involves significant expense. Heuristic approaches (Lee and El-Sharkawi, 2007) such as evolutionary algorithms (Branke et al., 2008; Deb et al., 2002; Kim et al., 2004), particle swarm optimization (Bonyadi and Michalewicz, 2017; Mason et al., 2017), and simulated annealing (Robert and Casella, 2004) avoid the need to know or approximate derivatives, but often require prohibitively many function evaluations. Furthermore, such methods, like gradient-based approaches, do not inherently provide quantification of remaining uncertainty about

optimal design settings and the system outputs at those settings. Methods exist for using heuristic approaches while accommodating and quantifying uncertainties (Deb and Gupta, 2006; Zhou et al., 2011), but these come at the cost of even further inflating the number of function evaluations required. This problem can be mitigated by relying on a surrogate model, but the resulting uncertainty quantification is accomplished by separate methods that are layered on top of the independent heuristic approach. On the other hand, our approach includes uncertainty quantification as an intrinsic aspect of the DCTO framework.

The third camp is the diverse collection of response surface methodologies (RSMs; Dean et al., 2017) used for optimization. RSMs operate by fitting a predictive model to an existing set of model runs, to form a computationally inexpensive meta-model which is then used to explore the model output. The concept of calibration to target outcomes that is built into DCTO is an example of an RSM, using GPs for its meta-model fit. Other popular versions of RSMs include efficient global optimization (EGO; Jones et al., 1998; Brochu et al., 2010) and stepwise uncertainty reduction (SUR; Geman and Jedynak, 1996; Villemonteix et al., 2009; Chevalier et al., 2014; Picheny, 2015; Miguel Hernández-Lobato et al., 2016; Picheny et al., 2019; Binois et al., 2019). EGO and SUR are both designed to facilitate sequential sampling from the system of interest in a search for the global optimum. They differ in their *acquisition functions*, which determine the location of the next sampling location throughout the optimization process. EGO finds the spot that maximizes the expected improvement (Mockus et al., 1978; Jones et al., 1998), whereas SUR's acquisition function seeks to reduce the volume of excursion sets below the current best known solutions (Chevalier et al., 2014). Because they rely on sequential sampling, EGO

and SUR are of limited utility when one is constrained to rely on a pre-existing set of observations, or in general when the observation locations cannot be chosen purely to suit the goal of optimization. Furthermore, the acquisition functions employed by EGO and SUR attempt to balance exploitation (proposing a new sample location that optimizes system output) with exploration (proposing a location that promotes learning for subsequent rounds of sampling). As a result, although these acquisition functions constitute distributions of sampling locations, by their nature they are not interpretable as distributions of the *optimal* design settings for a given problem, and hence these distributions do not quantify uncertainty regarding the location of that optimum. By contrast, our approach (understood as a pure-exploitation method) quantifies remaining uncertainty regarding the location of the system optimum.

An example of an RSM more closely resembling our approach to design is described by Olalotiti-Lawal and Datta-Gupta (2015). Their approach defines a distribution which is designed to lie both on and near the Pareto front (PF) of the objective function and generates a posterior distribution which includes quantified uncertainties via Markov chain Monte Carlo (MCMC; Gelfand and Smith, 1990). However, their posterior distribution is designed by the authors and is not dictated by the model itself; as such, its interpretability is not entirely clear. By contrast, our approach provides a posterior distribution based on the likelihood of the optimal design settings given the (hypothetical) observation of target outcomes y_t, and thus the uncertainty quantified by design using the KOH framework is model-driven and interpretable as uncertainty regarding the optimal values for the design inputs and the resulting system output.

The rest of the paper is organized as follows. Section 2 describes the difficulties involved in extending KOH into a framework that incorporates both design and calibration, illustrating this by considering the failings of a naïve method for combining the two procedures, followed by a description of the proposed DCTO framework for extending KOH. Section 3 showcases the methodology using a synthetic example, demonstrating its thorough quantification of the relevant uncertainties. Section 4 considers how DCTO may be useful in the case where sequential sampling is possible. In particular, sequential sampling with DCTO is attractive when the calibration parameter is known or suspected to be functionally dependent upon the design settings. Section 6 concludes with discussion of the results and thoughts about future directions. [1]

2 Dual calibration to target outcomes

The version of KOH considered here is that which finds a posterior distribution on a parameter of interest for calibration, $\boldsymbol{\theta}$, using a GP emulator with hyperparameters $\boldsymbol{\phi}_\eta$. Similarly, one may also use a GP prior with hyperparameters $\boldsymbol{\phi}_\delta$ to model discrepancy between the computer model $\eta()$ and the true function $f()$ that it represents. In the work described here, we employ stationary GPs with a Gaussian kernel covariance structure $C(\mathbf{x}, \mathbf{x}') = 1/\lambda \times \exp(-\beta(\mathbf{x} - \mathbf{x}')^2)$, so that $\boldsymbol{\phi}_\eta = [\beta,\ \lambda]$. In our adaptation, $\boldsymbol{\theta} = (\boldsymbol{\theta}_c, \boldsymbol{\theta}_d)$

[1] Note that KOH is usually conceived of as a means of calibrating a computer model with respect to a set of experimental observations. However, the KOH framework, and by extension DCTO, are applicable more generally whenever one has access to both low-fidelity and high-fidelity sources of information and seeks to calibrate the former with respect to the latter. This includes the case in which both the high-fidelity and low-fidelity sources of information are computer models (e.g. with different levels of computational expense). For ease of exposition, we follow the common convention, and present DCTO in terms of calibrating a computer model using experimental observations. Nonetheless, in some cases (such as in our discussion of sequential sampling in Section 4), the methods discussed may apply more naturally in the context of employing two computer models of varying fidelity.

is partitioned into parameters $\boldsymbol{\theta}_c$ to be calibrated and inputs $\boldsymbol{\theta}_d$ to be optimized for design purposes. Setting priors on $\boldsymbol{\theta}$ and on $\boldsymbol{\phi}_\delta$, we train the GP emulator on observations $\boldsymbol{\eta}$ and use MCMC to explore the distribution

$$\pi(\boldsymbol{\theta}, \boldsymbol{\phi}_\eta, \boldsymbol{\phi}_\delta | \mathcal{D}) \propto \pi(\mathcal{D} | \boldsymbol{\theta}, \boldsymbol{\phi}_\eta, \boldsymbol{\phi}_\delta) \times \pi(\boldsymbol{\theta}) \times \pi(\boldsymbol{\phi}_\eta) \times \pi(\boldsymbol{\phi}_\delta) \tag{1}$$

where $\mathcal{D} = (\boldsymbol{\eta}^T,\ \mathbf{y}^T)^T$ for some observations $\mathbf{y}$.

In a computer calibration problem, $\mathbf{y}$ is a set of observations of the system modeled by $\eta()$. When calibrating to target outcomes as in DCTO, by contrast, y is a set of target outcomes representing the way that one wishes to induce the system to behave (rather than observations one has made of the system in reality). When one wishes to perform design leveraging a simulation model that also requires traditional calibration, then, one might consider combining the two approaches by using Equation (1) with $\mathbf{y} = (\mathbf{y}_r^T,\ \mathbf{y}_t^T)^T$, an array containing both real observations $\mathbf{y}_r$ (for calibration) and target outcomes $\mathbf{y}_t$ (for design). However, this approach will not work, both because the inputs to be calibrated are typically not the same as the design settings under researcher control, and also because for successful calibration one must train one's model on observations of reality rather than on unobserved target outcomes.

Hence, model calibration and system design must be separated. An obvious choice here is to perform KOH calibration first, without involving any target outcomes, and then to use the calibrated model in order to perform CTO. Under this approach, with observations $\mathbf{y}_r$ of the system of interest, one would employ the model described in Equation (1) with $\boldsymbol{\theta} = \boldsymbol{\theta}_c$ (the parameters to be calibrated) and with $\mathcal{D} = \mathcal{D}_c = (\boldsymbol{\eta}^T, \mathbf{y}_r^T)^T$. The result

would be a posterior distribution of $\boldsymbol{\theta}_c$ and of $\delta(\cdot)$, the systematic discrepancy between the computer model $\eta(\cdot,\cdot)$ and the true system $f(\cdot)$. These can be used to produce estimates $\widehat{\boldsymbol{\theta}}_c$ and $\widehat{\delta}(\cdot)$ such that $f(\mathbf{z}) \approx \eta(\mathbf{z}, \widehat{\boldsymbol{\theta}}_c) + \widehat{\delta}(\mathbf{z})$ for all $\mathbf{z}$ in the domain of f. The result is a calibrated model $\eta_c(\mathbf{z}) = \eta(\mathbf{z}, \widehat{\boldsymbol{\theta}}_c) + \widehat{\delta}(\mathbf{z})$ which can be used for design.

With η_c in hand, one can partition $\mathbf{z}$ into $(\mathbf{x}, \boldsymbol{\theta}_d)$ where $\boldsymbol{\theta}_d$ is the set of inputs over which one wishes to optimize, and $\mathbf{x}$ are all other inputs in the operational domain, within which the calibrated model's predictions are reliable. We can write $\eta_c(\mathbf{z})$ as $\eta_c(\mathbf{x}, \boldsymbol{\theta}_d)$. Then one can perform design again using Equation (1), this time with $\boldsymbol{\theta} = \boldsymbol{\theta}_d$ and $\mathcal{D} = \mathcal{D}_t = (\boldsymbol{\eta}_c^T, \mathbf{y}_t^T)^T$ where $\boldsymbol{\eta}_c = \boldsymbol{\eta} + \widehat{\boldsymbol{\delta}} = \boldsymbol{\eta} + (\widehat{\delta}(\mathbf{z}_1), \cdots, \widehat{\delta}(\mathbf{z}_n))^T$. Notice that a single set of simulator runs $\boldsymbol{\eta}$ can be used both for KOH and for subsequent CTO. A crucial difference between calibration and design is that for the design step one would not attempt to model any systematic discrepancy between η_c and f, since an estimate of that discrepancy is already included in η_c. For the purposes of Equation (1), this amounts to setting a degenerate prior on ϕ_δ at 0.

A problem with the above-described approach of performing calibration prior to a separate design optimization is that relying on static calibration estimates $\widehat{\boldsymbol{\theta}}_c$ ignores uncertainty remaining after calibration with respect to the true value of $\boldsymbol{\theta}_c$. In order to produce results that take into account all sources of uncertainty, it is necessary to integrate calibration and design, so that the uncertainty remaining from calibration is propagated through the design process. This can be accomplished either asynchronously (so that the posterior distribution of $\widehat{\boldsymbol{\theta}}_c$ is sampled while undertaking design) or, for lower computational overhead, synchronously (so that a single MCMC run is used to perform both calibration and design).

In either case, it will be useful to produce an integrated model which describes the use of both procedures, and which makes clear the relationship between them. This integrated model will also serve to demonstrate the unified framework underlying the synchronous approach.

For this purpose, consider η as having three inputs $(\mathbf{x}, \mathbf{t}_c, \mathbf{t}_d)$ where $\mathbf{t}_c$ denotes the parameters targeted for KOH calibration, $\mathbf{t}_d$ denotes the input settings targeted for design, and $\mathbf{x}$ denotes the remaining controllable inputs. If η can be run quickly, then we use it directly in MCMC. However, if it is computationally expensive, we employ a surrogate by setting a Gaussian process (GP) prior on η with mean $m_\eta(\mathbf{x}, \mathbf{t}_c, \mathbf{t}_d)$ and covariance function $C_\eta((\mathbf{x}, \mathbf{t}_c, \mathbf{t}_d), (\mathbf{x}', \mathbf{t}_c', \mathbf{t}_d'))$. From here on in this discussion, assume that a GP surrogate is used for η. Model the systematic discrepancy between η and f at the true value of $\mathbf{t}_c = \boldsymbol{\theta}_c$ with another GP prior $\delta(\cdot, \cdot)$ having mean $m_\delta(\mathbf{x}, \mathbf{t}_d)$ and covariance function $C_\delta((\mathbf{x}, \mathbf{t}_d), (\mathbf{x}', \mathbf{t}_d'))$. In addition to systematic discrepancy between η and reality, measurement error ϵ_r may be included in the model for real observations $\mathbf{y}_r$, and additional Gaussian observation error ϵ_d may be included for target outcomes $\mathbf{y}_t$.

The purpose of additional observation error ϵ_d is twofold. Depending on the distribution of ϵ_c, the target outcomes $\mathbf{y}_t$ may or may not be possible outputs of a model that lacks ϵ_d. Including ϵ_d ensures that there is nonzero probability of an observation falling in the vicinity of the targets. Secondly, including ϵ_d and estimating its variance σ_d^2 provides computational benefits. For example, even if the target outcomes are compatible with a model that does not include ϵ_d, they may (depending on the choice of targets) be extreme outliers to the extent that the relevant likelihoods are small enough to generate significant

numerical errors during MCMC. In terms of the interpretation of the model, adding ϵ_d amounts to supposing that the counterfactual target outcomes were observed with greater than usual observation error, where that additional error is distributed as $N(0, \sigma_d^2)$. Though it is not necessary to assume that ϵ_c is Gaussian, for simplicity of presentation we assume here that it is distributed as $N(0, \sigma_c^2)$. Finally, we assume that η, δ, ϵ_c and ϵ_d are all mutually independent.

A collection of simulation runs is needed to train the GP code surrogate. Let $(\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}})$ be the design matrix for the settings of the simulation runs, and let $\mathbf{y_s}$ denote the output of these runs. Similarly, let $\mathbf{y_r}$ be observations made at $\mathbf{x_r}, \mathbf{t_{dr}}$, and let $\mathbf{y_t}$ be target outcomes we wish to observe at $\mathbf{x_t}$. Finally, let $\mathbf{y} = (\mathbf{y_s}^T, \mathbf{y_r}^T, \mathbf{y_t}^T)^T$, and $\mathbf{1}$ a vector of ones. Then it follows that $\mathbf{y} \sim \mathrm{N}(\mathbf{m}, \mathbf{C})$, where

$$\mathbf{m} = \begin{pmatrix} m_s(\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}) \\ m_s(\mathbf{x_r}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{t_{dr}}) + m_\delta(\mathbf{x_r}, \mathbf{t_{dr}}) \\ m_s(\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{1}\boldsymbol{\theta}_d^T) + m_\delta(\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_d^T) \end{pmatrix},$$

$$\mathbf{C} = \begin{pmatrix} \mathbf{C_{11}} & \mathbf{C_{12}} & \mathbf{C_{13}} \\ \mathbf{C_{21}} & \mathbf{C_{22}} & \mathbf{C_{23}} \\ \mathbf{C_{31}} & \mathbf{C_{32}} & \mathbf{C_{33}} \end{pmatrix},$$

$$\mathbf{C_{11}} = C_\eta\left((\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}), (\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}})\right)$$

$$\mathbf{C_{21}} = C_\eta\left((\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}), (\mathbf{x_r}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{t_{dr}})\right)$$

$$\mathbf{C_{31}} = C_\eta\left((\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}), (\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{1}\boldsymbol{\theta}_d^T)\right)$$

$$\mathbf{C_{12}} = \mathbf{C_{21}}^T$$

$$\mathbf{C_{22}} = C_\eta\left((\mathbf{x_r}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{t_{dr}}), (\mathbf{x_r}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{t_{dr}})\right) + C_\delta\left((\mathbf{x_r}, \mathbf{t_{dr}}), (\mathbf{x_r}, \mathbf{t_{dr}})\right) + \sigma_c^2\mathbf{I}$$

$$\mathbf{C_{32}} = C_\eta\left((\mathbf{x_r}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{t_{dr}}), (\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{1}\boldsymbol{\theta}_d^T)\right) + C_\delta\left((\mathbf{x_r}, \mathbf{t_{dr}}), (\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_d^T)\right)$$

$$\mathbf{C_{13}} = \mathbf{C_{31}}^T$$

$$\mathbf{C_{23}} = \mathbf{C_{32}}^T$$

$$\mathbf{C_{33}} = C_\eta\left((\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{1}\boldsymbol{\theta}_d^T), (\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{1}\boldsymbol{\theta}_d^T)\right) + C_\delta\left((\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_d^T), (\mathbf{x_t}, \mathbf{1}\boldsymbol{\theta}_d^T)\right) + \sigma_c^2\mathbf{I} + \sigma_d^2\mathbf{I}$$

Note that when $\mathbf{y_t}$ and $\mathbf{x_t}$ are empty and $\mathbf{m}, \mathbf{C}$ reduce respectively to their first two and upper two-by-two block elements, this is simply the KOH framework. Thus, DCTO is an extension of the KOH framework to include design using target outcomes.

A primary benefit of DCTO is that the design process includes quantification of all sources of uncertainty. Performing calibration and then subsequently undertaking design using static estimates for $\widehat{\boldsymbol{\theta}}_c$ and $\widehat{\delta}$ does not properly account for the uncertainty surrounding the estimates. Another benefit of the combined approach appears in cases in which $\boldsymbol{\theta}_c$ is suspected to be a function of $\boldsymbol{\theta}_d$. In such cases, one may be interested only or primarily in the value of $\boldsymbol{\theta}_c$ at the optimal value of $\boldsymbol{\theta}_d$. If one has the freedom to sample adaptively from the true system, then this freedom can be applied in DCTO to concentrate samples disproportionately in the region of interest. This idea is explored further in Section 4.

For DCTO, we employ modularity in the manner of Liu et al. (2009). A modular analysis intentionally falls short of being a full Bayesian analysis, either for computational benefits, or to quarantine "suspect" aspects of the model, so that the posterior distributions of parameters of interest are robust to model misspecification. The target outcomes $\mathbf{y}_t$ are precisely such a suspect source of Bayesian learning—they are by their nature extreme outliers, and hence are a poor guide both for estimating the hyperparameters of the GP emulator and for estimating the parameter $\boldsymbol{\theta}_c$. To modularize DCTO, we estimate the emulator hyperparameters via maximum likelihood, and we refrain from including $\mathbf{y}_t$ in the updates of $\boldsymbol{\theta}_c$ during MCMC. That is, rather than calculating the likelihood of a proposed sample $t_c^{(i+1)}$ at step i of the MCMC using $\mathbf{y} = (\mathbf{y}_s^T, \mathbf{y}_r^T, \mathbf{y}_t^T)^T$, we instead calculate its likelihood using only $\mathbf{y} = (\mathbf{y}_s^T, \mathbf{y}_r^T) \sim N(\mathbf{m}_r, \mathbf{C}_r)$, where $\mathbf{m}_r$ and $\mathbf{C}_r$ are respectively the upper two and upper-left two-by-two components of $\mathbf{m}$ and $\mathbf{C}$. Such modularization ensures that all Bayesian learning of $\boldsymbol{\theta}_c$ is based upon the real observations rather than upon $\mathbf{y}_t$.

3 Example with simulated data

Consider the

3.1 Results

In order to evaluate the success of the calibration component of DCTO, we also carried out a two-step procedure of using traditional KOH calibration of θ_c, followed by a second step using the KOH framework for design, obtaining a distribution of θ_d. The first step is essentially DCTO with $\mathbf{x_t}, \mathbf{y_t}$ as empty (null) vectors, and the second step uses the

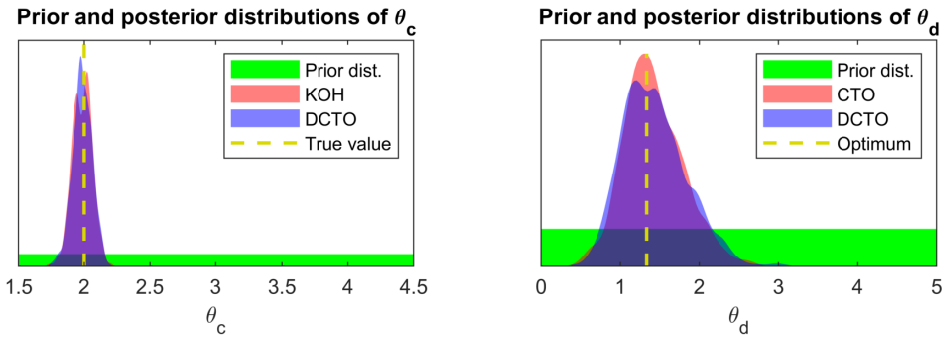

Figure 1: Prior and posterior distributions of the calibration parameter θ_c and design parameter θ_d, along with their true/optimal values, for DCTO and two-step calibration and design carried out when there is no discrepancy between the true system and the computer model. TODO WHICH DISCREP VERSION?

distributions obtained in the first step to estimate θ_d. Thus, the comparison between the unified approach, i.e. DCTO, and the two-step calibration and design approach shows the difference between DCTO and performing design on a system which has been calibrated using traditional methods and serves as validation of the former. Figure 1 shows the results of DCTO and two-step calibration and design for f_0, the case of no discrepancy. The two methods deliver comparable results, illustrating that extending calibration to include design does not undermine the performance of either task. Strong Bayesian learning has occurred for both parameters, in that the posterior distributions of θ_c, θ_d are peaked around their true and optimal values, respectively. KOH gives a similar posterior for θ_c, showing that the expansion of DCTO to undertake design has not interfered with its calibration performance. The skew apparent in the posterior distributions of θ_d occur in all of the results gathered here, and is likely due to the shape of the objective function f, which is much more informative below θ_d than above it in that it increases sharply for $t_d < \theta_d$ and increases much more gently for $t_d > \theta_d$.

	Posterior θ_c var.		Posterior θ_d var.	
Discrepancy	DCTO	KOH+design	DCTO	KOH+design
0	0.00633	0.00619	0.145	0.129
1, small	0.0140	0.0137	0.149	0.129
1, large	0.0141	0.0140	0.149	0.131
2, small	0.0143	0.0141	0.135	0.116
2, large	0.0609	0.0608	0.0804	0.0731
3, small	0.0134	0.0135	0.0882	0.0743
3, large	0.0143	0.0142	0.0945	0.0814

	$\widehat{\theta}_c$ RMSE		$\widehat{\theta}_d$ RMSE	
Discrepancy	DCTO	KOH+CTO	DCTO	KOH+CTO
0	0.0790	0.0795	0.120	0.121
1, small	0.0955	0.0956	0.149	0.144
1, large	0.137	0.139	0.209	0.209
2, small	0.109	0.106	0.130	0.127
2, large	0.158	0.155	0.123	0.121
3, small	0.294	0.292	0.0919	0.0919
3, large	0.279	0.281	0.0995	0.0990

Table 1: Posterior variance and root mean square error (RMSE) for the calibration variable θ_c and the design variable θ_d under both DCTO and two-step calibration and design (KOH+design). The estimator $\widehat{\theta}_i$ is the posterior mean of t_i for $i = c, d$.

We performed each procedure 30 times on each of the seven different discrepancy situations (no discrepancy, and a large and small version of each of three discrepancies). The results are summarized in Table 2. The upper table gives the sample mean, over the thirty runs, of the marginal posterior variance of each of θ_c and θ_d. The two procedures generate extremely similar outcomes with respect to θ_c. However, the posterior variance for θ_d under DCTO is slightly higher than that under two-step calibration and design in each of the seven cases considered. This is due to the fact that DCTO includes remaining uncertainty about the values of the hyperparameters of the discrepancy GP $\delta(\cdot)$. By contrast, design after calibration uses point estimates of those hyperparameters and thus achieves narrower posterior distributions due to excluding this source of uncertainty. The lower table gives

the root mean square errors (RMSEs) for the posterior means of θ_c and θ_d, using their true value of 2 for θ_c and optimal value 4/3 for θ_d in discrepancy cases $0, 1, 2$ and optimal value 1 for discrepancy 3. Again we see very similar outcomes in the two procedures for both parameters. In all cases but one, DCTO has slightly higher RMSE for θ_d than does design after KOH calibration. This is to be expected given the above-mentioned wider posterior distributions of θ_d under DCTO.

4 Dependence of θ_c on θ_d

In many cases of computer model calibration, it is known or suspected that the value of one or more calibration parameters are functionally dependent upon the values of other model inputs (Atamturktur and Brown, 2015; Atamturktur et al., 2017; Ezzat et al., 2018). If one is interested to understand the functional form of the calibration parameter, then state-aware methods can be used to arrive at such an estimate (Atamturktur and Brown, 2015; Atamturktur et al., 2017; Brown and Atamturktur, 2018).

In a case where the calibration parameter is functionally dependent upon the design settings, one might be interested only to know the value of the calibration parameter in the optimal design region. When calibration and design are undertaken simultaneously, as in DCTO, the machinery of state-aware calibration is not needed, and effort is better spent focusing on estimating the fixed calibration parameter value in the region of interest. In such a case, it is preferable that one's calibration be founded on observations for which the design settings are in the optimal design region. This will allow one to calibrate the model using observations taken from the region of design interest, so that the calibration takes

on values that are most applicable in that region.

When observations may be made adaptively, other RSM approaches such as EGO (Jones et al., 1998; Brochu et al., 2010) or SUR (Geman and Jedynak, 1996; Villemonteix et al., 2009; Chevalier et al., 2014; Picheny, 2015; Miguel Hernández-Lobato et al., 2016; Picheny et al., 2019; Binois et al., 2019) may be more efficient than the KOH framework for estimating optimal design settings, though the KOH framework offers more interpretable and model-driven uncertainty quantification. Further, RSM approaches in general do not include tools to accommodate the case in which a model stands in need of calibration as well as optimization. DCTO provides such a framework for combined calibration and design.

Therefore, we now consider under the lens of DCTO the case in which the design settings of the observations of the true system may be chosen adaptively. The use of DCTO with adaptive sampling is potentially of greatest use when it is known or suspected that the calibration parameter is a function $\theta_c(t_d)$ of the design setting t_d, and particularly when interest focuses on learning the optimal design setting θ_d and the corresponding value $\theta_c(\theta_d)$ of the calibration parameter. The process of performing DCTO with adaptive sampling is described in Algorithm 1. When adaptively evaluating the objective function, the locations

Algorithm 1: DCTO with adaptive sampling

1 Set $\mathbf{y} = [\mathbf{y}_r^T \mathbf{y}_t^T]^T$ where $\mathbf{y}_t$ are the target outcomes and $\mathbf{y}_r = [\ ]$ is an empty array.

2 Begin MCMC burn-in. Set $i = 1$. Let m be the budget of function evaluations. While $i \leq m$:

- 2.1 Complete n iterations of MCMC burn-in (where e.g. $n = 100$).
- 2.2 Draw $\widehat{\theta_d}$ from the available size $n \cdot i$ sample of $t_d|\mathbf{y}$.
- 2.3 Evaluate $f(\mathbf{x}_i, \widehat{\theta_d})$.
- 2.4 Set $\mathbf{y}_r = [\mathbf{y}_r^T f(\mathbf{x}_i, \widehat{\theta_d})]^T$.

3 Continue burn-in until convergence.

4 Draw a sample of desired size from the posterior distributions of θ_c, θ_d.

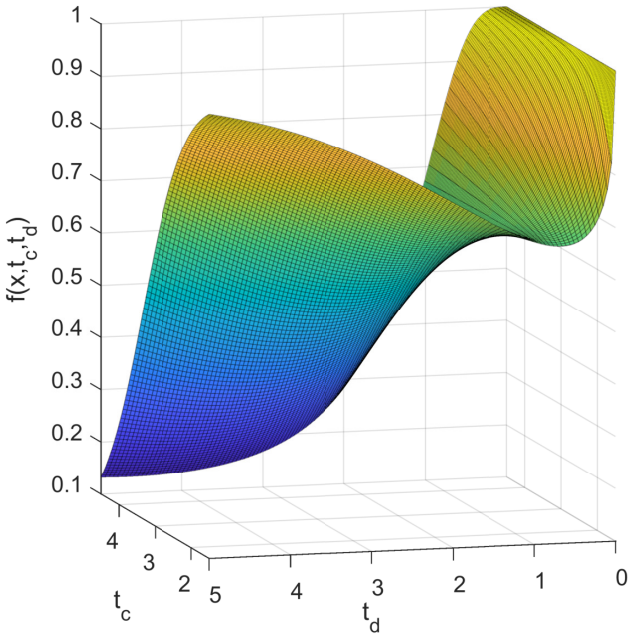

Figure 2: Example computer model output over the support of the calibration parameter t_c and the design parameter t_d.

of the input settings $\mathbf{x}_i$ which are not being optimized for design can be selected to maximize distance from previous observations, or these locations can be predetermined according to a space-filling design over the domain of non-design inputs. The result of applying this algorithm is that observations are concentrated around the design settings of interest, so that the unknown calibration parameter values in those observations are concentrated around the value $\theta_c(\theta_d)$.

To demonstrate the use of DCTO with adaptive sampling in a case of functional dependence of the calibration parameter on design settings, we use the function of three inputs $\eta(x, t_c, t_d) = x/(t_d^{t_c-1}\exp(-0.75t_d) + 1)$. Figure 2 shows the output of this function for $x = 1$ over the range $(t_c, t_d) \in [1.5, 4.5] \times [0, 5]$. For any value of x and t_c, the optimal (minimizing) value of t_d is $(4/3)(t_c - 1)$. Suppose that the calibration parameter's "true"

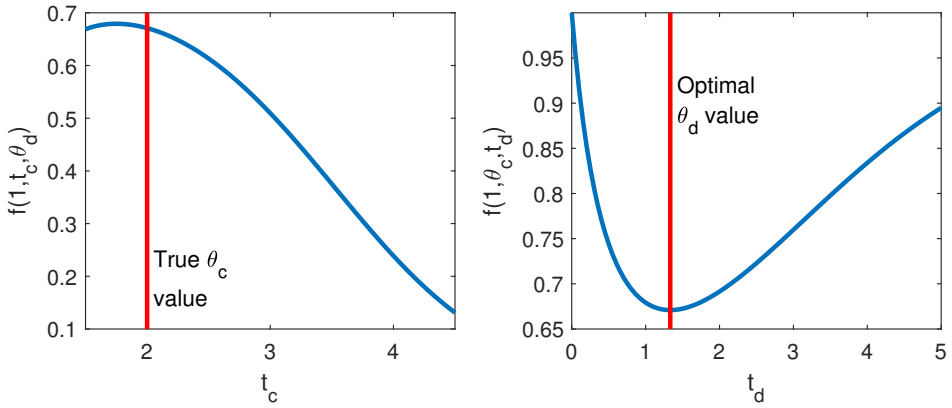

Figure 3: The lefthand plot shows the computer model output at $x = 1$ and optimal θ_d for each value of the calibration parameter t_c. The righthand plot show the model output at $x = 1, t_c = \theta_c$ for each value of the design parameter t_d.

value is functionally dependent on the design input, with the relationship:

$$\theta_c(t_d) = 2.25 - .75\frac{\exp\left(40\left(\frac{t_d-1.5}{.75} - .5\right)\right)}{1 + \exp\left(40\left(\frac{t_d-1.5}{.75} - .5\right)\right)}.$$

Figure **??** shows this relationship. Figure 3 shows the locations of the true and optimal values (respectively) of θ_c and θ_d. There it is clear that the true value of θ_c is far from optimal, in the sense that if this value *were* within our control (which, being a calibration parameter, it is not), we would prefer to place it at the upper end of its support, at 4.5. Thus η showcases the ability of DCTO to perform simultaneously both calibration and design in the case when our "truth-seeking" goals and our design goals are in tension.

We apply DCTO to four versions of the problem. First, we assume that η is free from discrepancy – i.e. that $\eta(x, \theta_c, t_d)$ is an unbiased estimator of the "true" system $f(x, t_d)$. The other three versions each assume that η suffers from some form of discrepancy. Let

f_1, f_2, f_3 denote the "true" systems in these three cases. We set

$$f_1(x, t_d) = \eta(x, \theta_c, t_d)\,(1 - a(x - .5)(x - 1)/x))$$
$$f_2(x, t_d) = \eta(x, \theta_c, t_d) - a(x - .5)(x - 1)\left(t_d - \frac{4}{3}\right)^2 + b$$
$$f_3(x, t_d) = \eta(x, \theta_c, t_d) + axt_d + b$$

where a, b are constants which determine how severe the discrepancy is in each case. The function f_1 has a multiplicative discrepancy dependent only on x and a. This discrepancy does not affect the optimal value of t_d. The discrepancies of f_2 and f_3 are both additive. Figure 4 shows the discrepancies for two different versions (corresponding to different settings of (a, b)) of each f_i.

We apply DCTO with and without adaptive sampling to each of seven cases, without using an emulator: the non-discrepancy case, and the two different versions of each f_i shown in Figure 4. In each case, we gather 30 "observations" of f_i on a latin hypercube design over the supports of x and t_d, setting θ_c equal to its "true" value of $\theta_c(t_d)$. After standardizing the response to have mean 0 and standard deviation 1, we add i.i.d. N(0,0.05) noise to the response. An example of the resulting "observations" from non-adaptive DCTO, with noise, appears in Figure 5. We carry out DCTO using Metropolis-Hastings-within-Gibbs MCMC, drawing 8000 realizations each (discarding the first 4000 as burn-in) of $t_c, t_d, \boldsymbol{\rho}_\delta, \lambda_\delta, \sigma_d^2$, where $\boldsymbol{\phi}_\delta = (\boldsymbol{\rho}_\delta^T, \lambda_\delta)^T$. For the adaptive sampling application of DCTO, we begin the MCMC with only TODO HOW MANY observations of f_i, making a new observation after every TODO HOW MANY steps of MCMC until we reached the total budget of 30. In

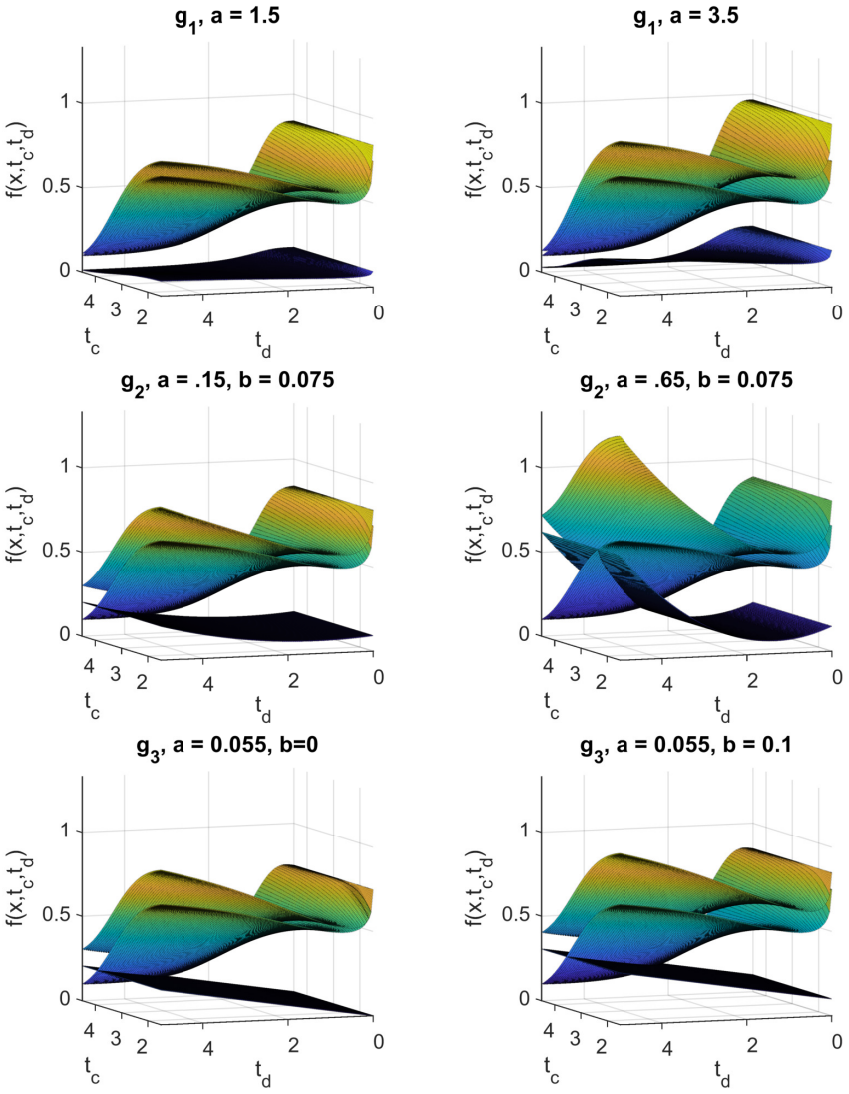

Figure 4: The i^{th} row shows f_i (the objective function with discrepancy), η (the computer model), and the discrepancy $f_i - \eta$, all at $x = 0.75$. In each row, a less aggressive version of the discrepancy appears on the left, and a more aggressive on the right. In each plot, the topmost surface is f_i, the middle surface is η, and the bottom surface is the discrepancy $f_i - \eta$.

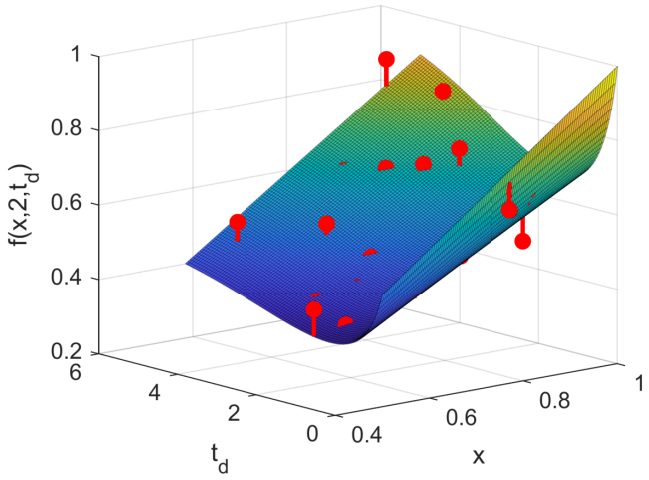

Figure 5: Noisy observations of the system, and the true system mean.

both versions of DCTO, we modularize the analysis by drawing each of $\boldsymbol{\theta}_c, \boldsymbol{\rho}_\delta, \lambda_\delta$ using the likelihood based only on $(\mathbf{y}_s^T, \mathbf{y}_r^T)^T$ rather than on $(\mathbf{y}_s^T, \mathbf{y}_r^T, \mathbf{y}_t^T)^T$. Convergence was verified visually and by the Gelman-Rubin statistic (≈ 1.01; Gelman and Rubin, 1992).

The resulting optimal design settings and calibration parameter value at the optimum vary in the discrepancy cases, though $\theta_c(\theta_d)$ is near 2.16 in each case. Representative results from performing DCTO with adaptive sampling in each discrepancy case appear in Figure 6, along with results from applying DCTO non-adaptively (using a space-filling set of observations). A summary of the results of thirty applications of DCTO both with and without adaptive sampling, for each of the discrepancy cases, appears in Table 2.

The results show superior performance for the adaptive sampling DCTO over DCTO using a space-filling design of experiments for the true phenomenon (or high-fidelity model, in a case of calibrating a low-fidelity model to use for design purposes). The adaptive DCTO posterior means have lower RMSEs in all cases for θ_d, and in all cases except one for θ_c. This demonstrates a useful robustness of adaptive DCTO to model misspecification,

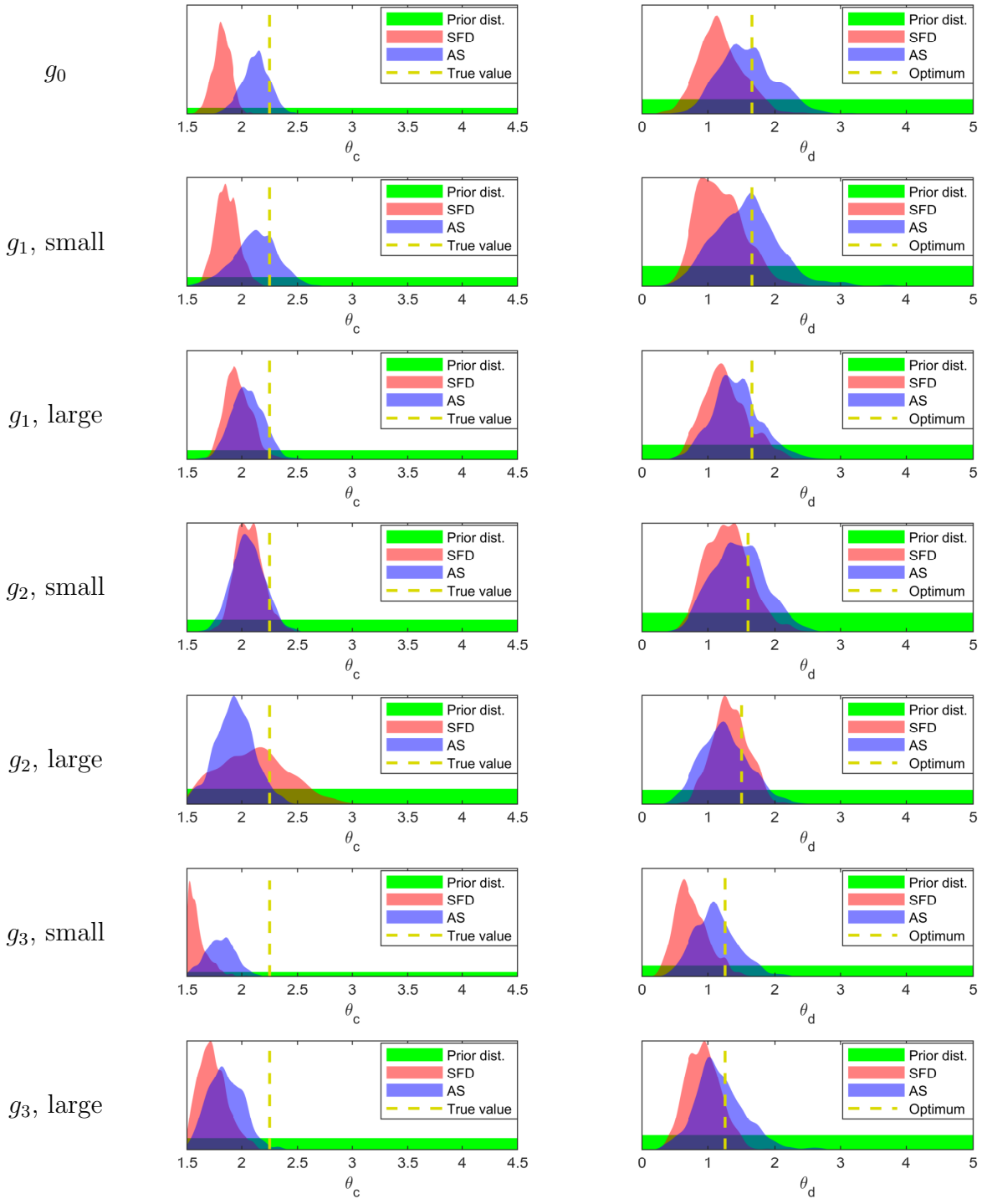

Figure 6: Prior and posterior distributions of the calibration parameter θ_c and design parameter θ_d, along with their true/optimal values, for DCTO with adaptive sampling (AS) and with predetermined space-filling design (SFD).

	$\widehat{\theta}_c$ RMSE		$\widehat{\theta}_d$ RMSE	
Discrepancy	AS	SFD	AS	SFD
0	0.188	0.433	0.163	0.479
1, small	0.233	0.32	0.243	0.414
1, large	0.188	0.247	0.213	0.393
2, small	0.221	0.263	0.187	0.348
2, large	0.228	0.16	0.183	0.206
3, small	0.452	0.506	0.182	0.329
3, large	0.448	0.468	0.167	0.292

Table 2: Posterior root mean square error (RMSE) for the calibration variable θ_c and the design variable θ_d, for DCTO with adaptive sampling (AS) and a predetermined space-filling design (SFD). The estimator $\widehat{\theta}_i$ is the posterior mean of t_i for $i = c, d$.

specifically in the case that the model treats as constant a calibration parameter that is more properly understood as functionally dependent upon other model inputs. By using the CTO-driven estimate $\widehat{\theta}_d$ to sample from the region of interest, DCTO learns from observations such that $\theta_c(\widehat{\theta}_d)$ is near to the value $\theta_c(\theta_d)$. This promotes better calibration with respect to the region of interest, and thereby better estimation of the optimal design settings. By relying on DCTO rather than on performing KOH using samples gathered using heuristic optimization methods, or other RSM approaches, we achieve these estimates with quantification of all relevant model-driven uncertainty with respect to the values of θ_c and θ_d.

5 Application: Dynamic vibration system

The dynamic vibration system (DVS) to which we apply DCTO is represented in Figure 7. In the DVS, an impulse force is applied to the frame of the system, causing the frame and the mass oscillator to accelerate. From the resulting acceleration over time of the mass

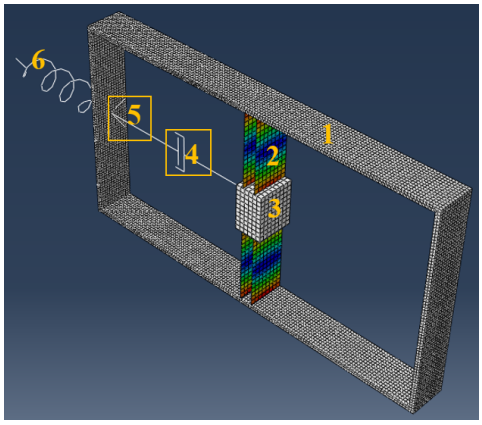

Figure 7: The dynamic vibration system. (1) The frame, (2) the beam, (3) the mass oscillator, (4) the spring, (5) the damper, and (6) the gain.

oscillator, we can calculate the damping ratio ζ of the system as

$$\delta = \frac{1}{n} \log \frac{x(t)}{x(t + nT)}, \quad \zeta = \frac{1}{\sqrt{1 + \left(\frac{2\pi}{\delta}\right)^2}}$$

where t is the time in seconds of the first peak value of mass acceleration, T is the time period of one oscillation, and n is the total number of time steps – i.e. the number of periods of the oscillation – used in the calculation. We use a finite element (FE) model of the system in conjunction with a set of 12 experimental observations in order to calibrate the elastic modulus of the spring. For the simulation data, $n = 1$, whereas for the experimental data we use $n = 4$ in order to account for measurement error TODO VERIFY THIS EXPLANATION. Due to the high computational cost of the FE model (roughly TODO HOW MANY seconds per model evaluation), we rely on a set of 98 runs for the calibration. In addition to the calibration, we undertake simultaneous design using these model runs and experimental observations in order to minimize the damping ratio of the system by tuning the value of the gain.

Thus in our application, the inputs of the FE system are the mass m of the oscillator (in kilograms), the elastic modulus k of the spring (in Newtons per meter), and the gain g (in Newtons per meter per second). The relevant output of the system is the damping ratio. The simulator is implemented in the commercial finite element software ABAQUS (Simulia, 2012) TODO VERIFY CORRECT VERSION. The experimental design for the simulator runs is a latin hypercube over a range of the inputs that covers those of the available experimental data for the known inputs m and g, and that covers a plausible range of values for the calibration input k. The experimental design is a full factorial design over three different mass levels (0.7853, 0.9653, and 1.1493kg) and four different gain levels (0, 8, 41, and 95 Ns/m). The expected elastic modulus for the system is 6.2e+10, as estimated by the leaf spring manufacturer.

Since our goal is to minimize the damping ratio, we set our target outcomes y_t to be 0 across a range of oscillator masses. Specifically, we set a grid of size 8 over the range of oscillator masses present in the simulation and experimental data, with target outcome 0 for each point in that grid. We define our prior GP surrogate for the FE model using a mean function found via degree-2 polynomial regression on the available FE runs. For the hyperparameters of the surrogate's covariance function, we estimate them as MLEs using the quasi-Newton BFGS method (Fletcher, 2013). We perform 10,000 iterations of MCMC using this surrogate and set of target observations, of which the first half are discarded as burn-in. The convergence of the resulting MCMC chains is assessed both visually and using the Gelman-Rubin statistic (≈ 1.01 and 1.001 for calibration and design respectively); Gelman and Rubin, 1992).

The total wall time required for the MCMC to complete DCTO in this case was 94 seconds (on a laptop with an Intel Core i7-9750H CPU and 16GB of RAM). The posterior distributions of the calibration and design inputs are are shown in Figure **??**. In comparison with the uniform prior distributions of the inputs, strong Bayesian learning has occurred, particularly for the design input. The posterior distribution of the elastic modulus for the system assigns high likelihood to the expected value of 6.2e10, with a posterior mean of 6.188e10. We use the surrogate model to estimate also the posterior predictive distribution of the system after DCTO. Figure **??** shows the resulting posterior distributions of model output at various levels of oscillator mass, along with the distributions of both experimental and simulator system output. Note that the predicted model outputs fall at the bottom of the ranges of observed model outputs across the domain of oscillator masses, implying a successful design outcome for the system has been achieved.

6 Conclusion

DCTO provides a method for generalizing the KOH framework for model calibration to include design. The result secures the benefits of KOH both for calibration and for design. This includes the ability to quantify uncertainty remaining in the true value of the calibration parameter, the optimal settings for the design input, and the resulting model output. DCTO achieves similar results to performing KOH calibration followed by design optimization, but provides a computationally efficient method of propagating the uncertainties remaining from KOH calibration through the design procedure. In the case when observations of the real system can be carried out sequentially at adaptively chosen locations,

DCTO is robust to model misspecification where the calibration parameter is functionally dependent on the value of the design input, and the model fails to reflect this. In such a case, if the functional form of the dependence of $\boldsymbol{\theta}_c$ on $\boldsymbol{\theta}_d$ is of interest, then state-aware calibration should be used. However, if one only wishes to estimate the calibration parameter at the optimal design settings, then DCTO provides a means of doing so. In this application, DCTO with adaptive sampling uses information from both the sequentially-performed observations of the real system and from the existing computer model to identify new sampling locations. Future work on this subject will include pairing adaptive sampling DCTO with other methodologies for selecting new sampling locations, such as EGO and SUR.

References

Atamturktur, S. and D. A. Brown (2015). State-aware calibration for inferring systematic bias in computer models of complex systems. *NAFEMS World Congress Proceedings, June 21-24*.

Atamturktur, S., J. Hegenderfer, B. Williams, M. Egeberg, R. A. Lebensohn, and C. Unal (2017). Mechanics of advanced materials and structures: a resource allocation framework for experiment-based validation of numerical models. *Mechanics of Advanced Materials and Structures 22*(8), 641–654.

Bayarri, M. J., J. O. Berger, and J. Cafeo (2007a). Computer model validation with functional output. *The Annals of Statistics 35*, 1874–1906.

Bayarri, M. J., J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu (2007b). A framework for validation of computer models. *Technometrics 49*(2), 138–154.

Binois, M., V. Picheny, P. Taillandier, and A. Habbal (2019, feb). The Kalai-Smorodinski solution for many-objective Bayesian optimization.

Bonyadi, M. R. and Z. Michalewicz (2017, mar). Particle swarm optimization for single objective continuous space problems: A review.

Branke, J., K. Deb, K. Miettinen, and R. Slowinski (Eds.) (2008). *Multiobjective Optimization: Interactive and Evolutionary Approaches*, Volume 5252 LNCS. Springer.

Brochu, E., V. M. Cora, and N. de Freitas (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.

Brown, D. A. and S. Atamturktur (2018). Nonparametric functional calibration of computer models. *Statistica Sinica 28*(2), 721–742.

Brynjarsdóttir, J. and A. O'Hagan (2014). Learning about physical parameters: The importance of model discrepancy. *Inverse Problems 30*(11).

Chevalier, C., J. Bect, D. Ginsbourger, E. Vazquez, V. Picheny, and Y. Richet (2014). Fast Parallel Kriging-Based Stepwise Uncertainty Reduction With Application to the Identification of an Excursion Set. *Technometrics 56*(4).

Dean, A., D. Voss, and D. Draguljić (2017). *Response Surface Methodology*, pp. 565–614. Cham: Springer International Publishing.

Deb, K. and H. Gupta (2006, dec). Introducing robustness in multi-objective optimization. *Evolutionary Computation 14*(4), 463–494.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*(2), 182–197.

Ezzat, A. A., A. Pourhabib, and Y. Ding (2018, jul). Sequential Design for Functional Calibration of Computer Models. *Technometrics 60*(3), 286–296.

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

Gelfand, A. E. and A. F. M. Smith (1990, jun). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association 85*(410), 398–409.

Gelman, A. and D. B. Rubin (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science 7*(4), 457–472.

Geman, D. and B. Jedynak (1996). An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*(1), 1–14.

Higdon, D., M. Kennedy, J. C. Cavendish, J. A. Cafeo, and R. D. Ryne (2004). Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing 26*(2), 448–466.

Jones, D. R., M. Schonlau, and W. J. Welch (1998). Efficient Global Optimization of Expensive Black-Box Functions. Technical report.

Kennedy, M. C. and A. O'Hagan (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B 63*(3), 425–464.

Kim, M., T. Hiroyasu, M. Miki, and S. Watanabe (2004). SPEA2+: Improving the performance of the strength pareto evolutionary algorithm 2. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3242*, 742–751.

Lee, K. Y. and M. A. El-Sharkawi (2007, jun). *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. John Wiley and Sons.

Liu, F., M. J. Bayarri, and J. O. Berger (2009). Modularization in Bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis 4*(1), 119–150.

Mason, K., J. Duggan, and E. Howley (2017, dec). Multi-objective dynamic economic emission dispatch using particle swarm optimisation variants. *Neurocomputing 270*, 188–197.

Miguel Hernández-Lobato, J., M. A. Gelbart, R. P. Adams, M. W. Hoffman, Z. Ghahramani, J. M. Hernández-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani Hernández-Lobato (2016). A General Framework for Constrained Bayesian Optimization using Information-based Search. Technical report.

Mockus, J., V. Tiesis, and A. Zilinskas (1978). The application of bayesian methods for seeking the extremum. *Towards global optimization 2*(117-129), 2.

Nocedal, J. and S. J. Wright (2006). *Numerical optimization*. Springer.

Olalotiti-Lawal, F. and A. Datta-Gupta (2015, sep). A Multi-Objective Markov Chain Monte Carlo Approach for History Matching and Uncertainty Quantification. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

Paulo, R., G. García-Donato, and J. Palomo (2012). Calibration of computer models with multivariate output. *Computational Statistics and Data Analysis 56*, 3959–3974.

Picheny, V. (2015). Multiobjective optimization using Gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing 25*(6), 1265–1280.

Picheny, V., M. Binois, and A. Habbal (2019, jan). A Bayesian optimization approach to find Nash equilibria. *Journal of Global Optimization 73*(1), 171–192.

Regis, R. G. and C. A. Shoemaker (2004). Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation 8*(5), 490–505.

Robert, C. P. and G. Casella (2004). Monte Carlo Optimization. In *Monte Carlo Statistical Methods* (2 ed.)., Chapter 5, pp. 157–204. New York, NY: Springer New York.

Simulia, D. S. (2012). Abaqus 6.12 documentation. *Providence, Rhode Island, US 261*.

Villemonteix, J., E. Vazquez, and E. Walter (2009, aug). An informational approach to

the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization 44*(4), 509–534.

Williams, B., D. Higdon, J. Gattiker, L. Moore, M. McKay, and S. Keller-McNulty (2006). Combining experimental data and computer simulations, with an application to flyer plate experiments. *Bayesian Analysis 1*(4), 765–792.

Zhou, A., B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang (2011, mar). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation 1*(1), 32–49.