Combining model calibration and design

Carl Ehrett*
School of Mathematical and Statistical Sciences, Clemson University,
D. Andrew Brown
School of Mathematical and Statistical Sciences, Clemson University,
Evan Chodora
Department of Mechanical Engineering, Clemson University,
Christopher Kitchens
Department of Chemical and Biomolecular Engineering, Clemson University,
and
Sez Atamturktur
Department of Architectural Engineering, Pennsylvania State University

Keywords:

*The authors gratefully acknowledge *please remember to list all relevant funding sources in the unblinded version*

1 Introduction

The goal of traditional Kennedy-O'Hagan style calibration (KOH, Kennedy and O'Hagan, 2001) is to find a posterior distribution on unknown parameters by calibrating a computer model using real-world observations of the modeled phenomenon. By contrast, the design methodology of calibration to target outcomes (CTO) uses the KOH framework to find a posterior distribution on optimal input settings in the model by "calibrating" a computer model using artificial observations that reflect performance and cost targets for the modeled system. The goal of the work described here is to combine KOH and CTO. Call the resulting methodology DCTO, for dual calibration to target outcomes.

CTO as previously developed assumes, somewhat idealistically, that the computer model is already perfectly calibrated. DCTO avoids this idealization. Furthermore, when undertaking KOH, some areas of the model range may be of greater interest than others. For example, one may be more interested in calibrating the model to be accurate in the optimal region of some design variable θ than elsewhere. Undertaking dual calibration may allow us to focus our calibration efforts on such regions of interest, prioritizing them over other areas of the model range.

2 Naïve DCTO

The version of KOH considered here is that which finds a posterior distribution on a parameter of interest, $\boldsymbol{\theta}$, using a GP emulator with hyperparameters $\widehat{\boldsymbol{\phi}_\eta}$ estimated via maximum likelihood using a budget of computer model observations $\boldsymbol{\eta}$. Similarly, we use

a GP prior with hyperparameters ϕ_δ 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 $\widehat{\boldsymbol{\phi}_\eta} = \left[\widehat{\beta},\ \widehat{\lambda}\right]$. Setting priors on $\boldsymbol{\theta}$ and on ϕ_δ, we train the GP emulator on observations $\boldsymbol{\eta}$ and use MCMC to explore the distribution

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

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

In a computer calibration problem, $\mathbf{y}$ is a set of observations of the system modeled by $\eta()$. In CTO, by contrast, y is a set of target outcomes – artificial data 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 CTO on a system that also requires traditional calibration, then, one obvious idea is to combine 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 CTO). However, this approach will not work, for two reasons. Firstly, the inputs $\boldsymbol{\theta}$ are typically not the same in calibration and in CTO. In calibration, one seeks to estimate the value of an input that either represents some true unknown quantity, or else which induces the model to represent some true unknown quantity. In CTO, one seeks to find a distribution on an input that is under the researcher's control.

Relatedly, by including target outcomes in one's observations $\mathbf{y}$ used in KOH calibration, one compromises the integrity of the calibration. KOH calibration, after all, aims to use

one's observations of the real system to estimate the value of $\boldsymbol{\theta}$. If one's observations $\mathbf{y}$ do not reflect the behavior of the real system, then they will constitute a poor source of information for bringing the model into alignment with reality. Simply put, to represent reality one must train one's model on observations of reality, not on unobserved targets.

3 Dual calibration to target outcomes

Given that naïve DCTO cannot accomplish the dual tasks of calibration and design, a different approach must be taken. The two tasks 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 $\mathcal{D} = \mathcal{D}_c = (\boldsymbol{\eta}^T, \mathbf{y}_c^T)^T$. 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 CTO.

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 to the calibrated model. We can write $\eta_c(\mathbf{z})$ as $\eta_c(\mathbf{x}, \boldsymbol{\theta}_d)$. Then one can perform CTO again using Equation (1), this time with $\boldsymbol{\theta} = \boldsymbol{\theta}_d$ and $\mathcal{D} = \mathcal{D}_d = (\boldsymbol{\eta}_c^T, \mathbf{y}_d^T)^T$ 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 KOH and CTO is that for the CTO 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.

Above, the use of CTO following KOH relies on two separate implementations of Equation (1). It will be useful to produce an integrated model which describes the use of both procedures, and which makes clear the relationship between them. 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 via CTO, and $\mathbf{x}$ denotes the remaining known and/or 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 $C_0((\mathbf{x}, \mathbf{t}_c, \mathbf{t}_d), (\mathbf{x}', \mathbf{t}_c', \mathbf{t}_d'))$. 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 $\mathbf{y}_c$, and additional Gaussian observation error ϵ_d may be included for target outcomes $\mathbf{y}_d$. The purpose of additional observation error ϵ_d is twofold. Depending on the distribution of ϵ_c, the target outcomes $\mathbf{y}_d$ may or may not be within the support of a model that lacks ϵ_d. Including ϵ_d ensures that the targets are compatible with the model. 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 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 observations made in ω were subject to 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 these simulation runs, and let $\mathbf{y_s}$ denote the output of these runs. Similarly, let $\mathbf{y_c}$ be observations made at $\mathbf{x_c}, \mathbf{t_c}$, and let $\mathbf{y_d}$ be target outcomes "observed" at $\mathbf{x_d}$. Finally, let $\mathbf{y} = (\mathbf{y_s}^T, \mathbf{y_c}^T, \mathbf{y_d}^T)^T$. Then it follows that $\mathbf{y} \sim \mathrm{N}(\mathbf{m}, \mathbf{C})$, where

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

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

$$\mathbf{C_{11}} = C_0\left((\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}), (\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}})\right)$$

$$\mathbf{C_{21}} = C_0\left((\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}), (\mathbf{x_c}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{t_{dc}})\right)$$

$$\mathbf{C_{31}} = C_0\left((\mathbf{x_s}, \mathbf{t_{cs}}, \mathbf{t_{ds}}), (\mathbf{x_d}, \mathbf{1}\boldsymbol{\theta}_c^T, \mathbf{1}\boldsymbol{\theta}_d^T)\right)$$

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

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

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

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

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

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

Note that when $\mathbf{y_d}$ and $\mathbf{x_d}$ 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, this combination of KOH and CTO generalizes the KOH framework.

This inclusive framework allows for KOH and CTO to be undertaken simultaneously. Call this inclusive framework DCTO, for dual calibration with target outcomes. A primary benefit of DCTO is that under this combined approach, the results of CTO include quantification of all sources of uncertainty. By performing KOH and then subsequently undertaking CTO using static estimates $\widehat{\boldsymbol{\theta}_c}$ and $\widehat{\delta}$ from KOH, uncertainty surrounding those estimates is not included in the results of CTO. 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 XXX.

Another crucial difference between performing CTO after KOH and performing DCTO is the role of the targets $\mathbf{y}_d$ in the calibration of $\boldsymbol{\theta}_c$. In DCTO as described above, the likelihood of $\boldsymbol{\theta}_c$ is affected by $\mathbf{y}_d$, whereas in KOH $\mathbf{y}_d$ is not included in the model and hence cannot affect the distribution of $\boldsymbol{\theta}_c$. This is a point in favor of performing KOH separate from CTO, as $\mathbf{y}_d$ is artificial data and cannot plausibly serve as a source of information about the correct value of $\boldsymbol{\theta}_c$. A similar issue affects CTO when an emulator is used. In KOH, the hyperparameters of an emulator may be estimated prior to calibration (e.g. via maximum likelihood), or else one may set priors for these hyperparameters and sample from their posteriors during the calibration process. If the latter route is chosen for CTO, then one would again face a situation in which learning about real quantities from artificial data. The hyperparameters of the GP emulating η should be estimated using observations of η and of the real process that η simulates. The solution in the case of CTO is to employ some form of modularity (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. The target outcomes $\mathbf{y}_d$ 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 $\boldsymbol{\theta}_c$. In CTO, modularization with respect to the GP hyperparameters typically takes the form of producing maximum likelihood estimates of the GP hyperparameters and using those in

lieu of setting priors and exploring a posterior distribution. Modularization with respect to $\boldsymbol{\theta}_c$ is implicit in the fact that an estimate $\widehat{\boldsymbol{\theta}_c}$ is used in CTO after KOH. DCTO can similarly modularize simply by refraining from including $\mathbf{y}_d$ 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 $\mathbf{y} = (\mathbf{y}_s^T, \mathbf{y}_c^T, \mathbf{y}_d^T)^T$, one can instead calculate its likelihood using only $\mathbf{y}_r = (\mathbf{y}_s^T, \mathbf{y}_c^T) \sim N(\mathbf{m}_r, \mathbf{C}_r)$, 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 $\mathbf{y}_d$.

4 Example with simulated data

Consider the function of three inputs $\eta(x, t_c, t_d) = x/(t_d^{t_c - 1} \exp(-0.75 t_d) + 1)$. Figure 1 shows the output of this function for $x = 1$ over the range $(t_c, t_d) \in [1.5, 4.5] \times [0, 5]$. We arbitrarily set $\theta_c = 2$ to be the "true" value of t_c. For any value of x and t_c, the optimal (minimizing) value of t_d is $(4/3)(t_c - 1)$, so we have $\theta_d = 4/3$. Figure 2 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 – if this value were within our control, its optimal value would be 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.

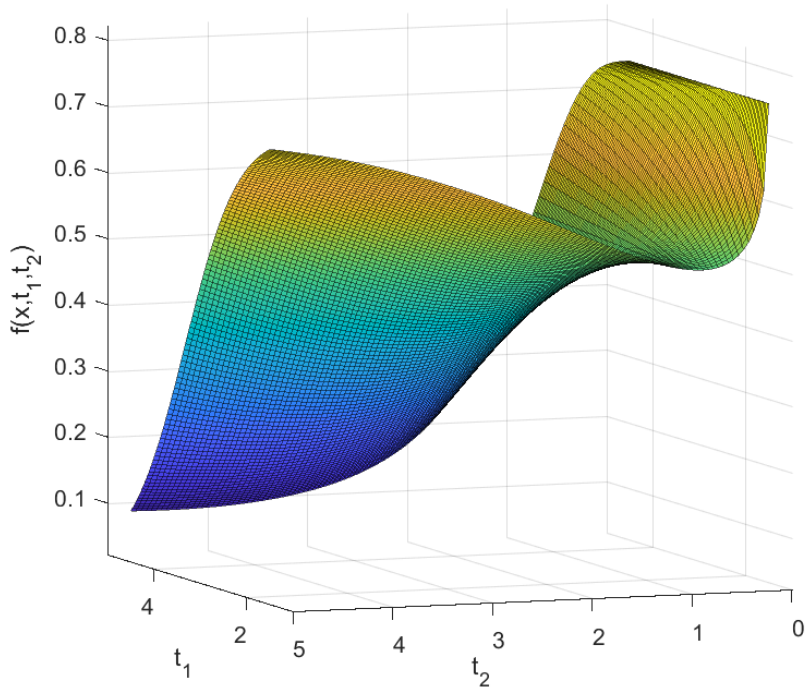

Figure 1: Example computer model output over the support of the calibration parameter t_c and the design parameter t_d.

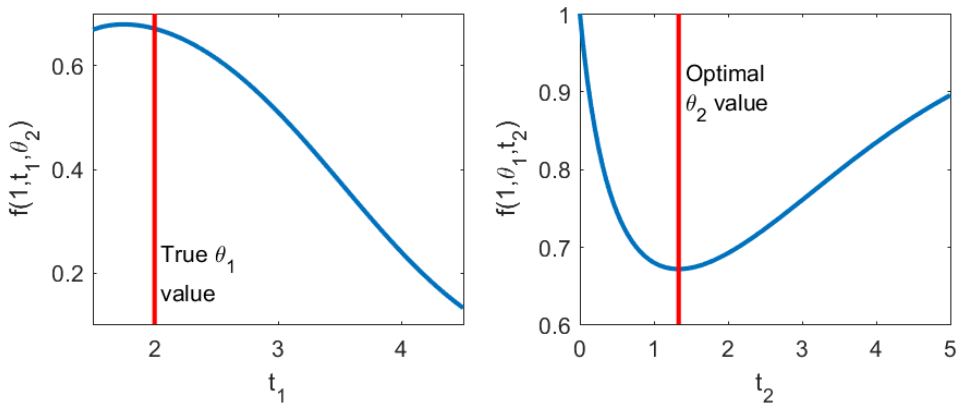

Figure 2: The lefthand plot shows the computer model output at $x = 1$ and optimal θ_d for each value of the calibration parameter t_c. The righthand plot show the model output at $x = 1, t_c = \theta_c$ for each value of the design parameter t_d.

4.1 Results

We used DCTO on four versions of the problem. First, we assumed 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. This discrepancy does not affect the optimal value of t_d. The discrepancy of f_2 is additive, and is dependent upon both x and θ_c. Though this discrepancy can affect the optimal value of t_d, in the case that $\theta_c = 2$ (which is what we assume to be the truth) it does not. Thus under f_1 and f_2, it remains the case that the optimal value of t_d is $\theta_d = 4/3$. By contrast, f_3 has an additive discrepancy which does affect the optimal setting for t_d. For f_3, optimal t_d is dependent upon both the true value of θ_c and upon the value of a. For example, for $\theta_c = 2$ and $a = 0.055$, the optimal t_d is $\theta_d \approx 1$. Figure 3 shows the discrepancies for two different versions (corresponding to different settings of (a, b)) of each f_i.

We applied DCTO to each of seven cases: the non-discrepancy case, and the two different versions of each f_i shown in Figure 3. We found that in these cases, no appreciable

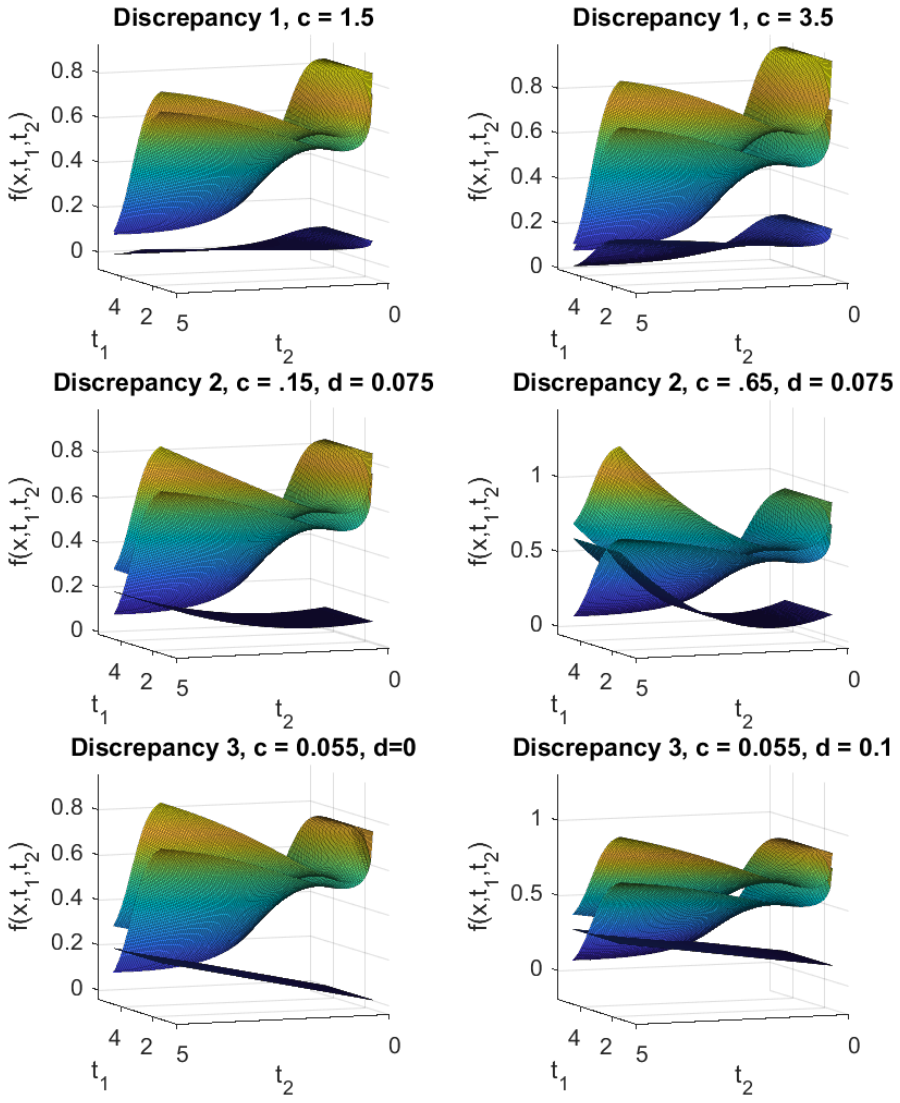

Figure 3: The i^{th} row shows f_i (the objective function with discrepancy), η (the computer model), and the discrepancy $f_i - \eta$, all at $x = 0.75$. In each row, a less aggressive version of the discrepancy appears on the left, and a more aggressive on the right. In each plot, the topmost surface is f_i, the middle surface is η, and the bottom surface is the discrepancy $f_i - \eta$.

difference resulted from the decision of whether or not to use an emulator (where the emulator was trained on a latin hypercube design of 250 points on the space of model inputs). Therefore, the results reported here do not employ an emulator. In each case, we gathered 50 "observations" of f_i on a latin hypercube design over the supports of x and θ_d, setting θ_c equal to its "true" value of 2. After standardizing the response to have mean 0 and standard deviation 1, we added i.i.d. N(0,0.05) noise to the response. We then carried out DCTO using Metropolis-Hastings-within-Gibbs MCMC, drawing 8000 samples each of $t_c, t_d, \boldsymbol{\rho}_\delta, \lambda_\delta, \sigma_d^2$, where $\boldsymbol{\phi}_\delta = (\boldsymbol{\rho}_\delta^T, \lambda_\delta)^T$. We modularized the analysis by drawing each of $\boldsymbol{\theta}_c, \boldsymbol{\rho}_\delta, \lambda_\delta$ using the likelihood based only on $\mathbf{y}_r$ rather than on $\mathbf{y}$.

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 CTO to obtain a distribution of θ_d. The first step is essentially DCTO with $\mathbf{x_d}, \mathbf{y_d}$ as empty (null) vectors, and the second step uses the distributions obtained in the first step to estimate θ_d. Thus, the comparison between DCTO and KOH+CTO shows the difference between DCTO and performing CTO on a system which has been calibrated using traditional methods. Figure 4 shows the results of DCTO and KOH+CTO for f_0, the case of no discrepancy. The two methods deliver comparable results, illustrating that combining KOH calibration and CTO design into DCTO does not undermine the performance of either task. Strong Bayesian learning has occurred for both parameters, in that the posterior distributions of θ_1, θ_2 are peaked around their true and optimal values, respectively. KOH gives a similar posterior for θ_1, showing that the expansion of DCTO to undertake design has not interfered with its calibration performance. The skewness apparent in the posterior

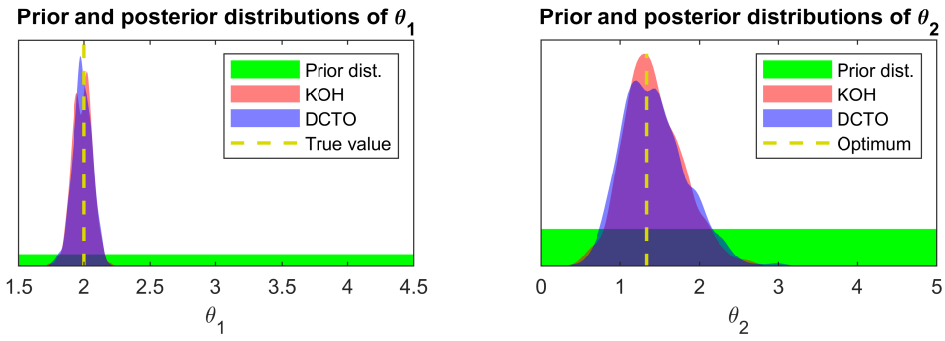

Figure 4: Prior and posterior distributions of the calibration parameter θ_1 and design parameter θ_2, along with their true/optimal values, for DCTO and KOH+CTO carried out when there is no discrepancy between the true system and the computer model.

distributions of θ_2 occur in all of the results gathered here, and is likely due to the shape of the objective function f, which increases sharply for $t_2 < \theta_2$ and increases much more gently for $t_2 > \theta_2$.

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 CTO 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 $\delta()$. By contrast, CTO after KOH 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 $theta_d$ in discrepancy cases $0-2$ and 1 for discrepancy 3. Again we see

	Posterior θ_c var.		Posterior θ_d var.	
Discrepancy	DCTO	KOH+CTO	DCTO	KOH+CTO
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_1 and the design variable θ_2 under both DCTO and KOH+CTO. The estimator $\widehat{\theta}_i$ is the posterior mean of t_i.

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 CTO. This is to be expected given the above-mentioned wider posterior distributions of θ_d under DCTO.

5 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). However, in a case of simultaneous calibration and design in which 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. In this case, 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 interest, so that the calibration takes on values from that region.

Recall that 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 CTO for estimating optimal design settings, though CTO offers more interpretable and model-driven uncertainty quantification. However, 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.

An obvious approach when both calibration and design are undertaken and the calibration parameter is suspected to be functionally dependent on the design settings is to perform the design optimization first, and then the calibration. In this way, one would gather observations in service of finding the optimal design settings, and then use these observations to learn the value of the calibration parameter in the associated optimal design region. For this purpose, any optimization method could be used to select the design settings for the observations. If a limited budget of observations is available and quantification of remaining uncertainty surrounding the true and optimal values (respectively) of the calibration parameter and design settings is desired, then RSMs are a good choice. In such a situation, CTO retains its advantage of interpretable and model-driven uncertainty quantification, and adds another advantage: namely, that DCTO constitutes a method for combining both the observations of the true system and the existing uncalibrated computer model to select new evaluation points.

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. To implement DCTO with adaptive sampling in such a case, we perform DCTO beginning with an empty vector of true observations y_r. During the burn-in period, every (e.g.) 100 iterations of the MCMC, we sample draw a sample $\widehat{\theta}_d$ from the posterior distribution of the design settings t_d, as a means of estimating the optimal design settings θ_d. We then perform a new observation of the system at $(\mathbf{x}_i, \widehat{\theta}_d)$ where $i \in \{1, \cdots, m\}$ and m is the total budget of observations to be made. The non-design input settings $\mathbf{x}_i$ can be chosen to maximize distance from previous observations, or these locations can be predetermined according to a space-filling design over the domain $\mathcal{X}$ of non-design inputs. This process is continued until the total budget of observations is reached, after which burn-in continues until the MCMC chains converge for all parameters of interest. The result 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 same objective function and discrepancy cases as described in Section 4. This time, instead of setting $\theta_c = 2$, we set

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

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

	$\widehat{\theta}_c$ 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_1 and the design variable θ_2, 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.

5, 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. 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, 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.

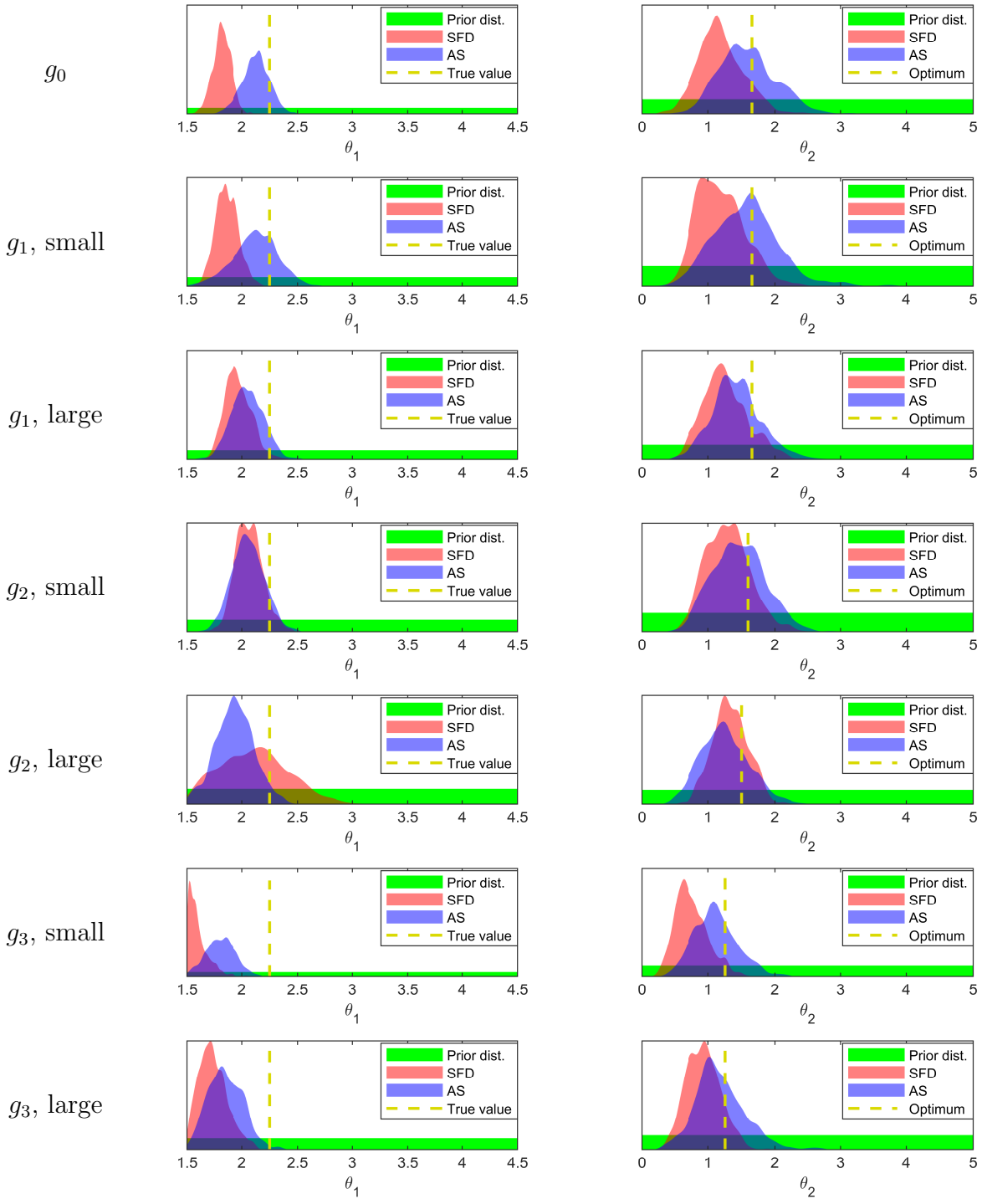

Figure 5: Prior and posterior distributions of the calibration parameter θ_1 and design parameter θ_2, along with their true/optimal values, for DCTO with adaptive sampling (AS) and with predetermined space-filling design (SFD).

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.

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

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.

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).

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

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.

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.

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.

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.

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.

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.