LA-UR-18-21836 (Accepted Manuscript)

Inferring Atmospheric Release Characteristics in a Large Computer Experiment using Bayesian Adaptive Splines

Francom, Devin Craig
Sanso, B
Bulaevskaya, V
Lucas, D
Simpson, M

To be published in: Journal of the American Statistical Association

DOI to publisher's version: 10.1080/01621459.2018.1562933

Permalink to record: http://permalink.lanl.gov/object/view?what=info:lanl-repo/lareport/LA-UR-18-21836

Inferring Atmospheric Release Characteristics in a Large Computer Experiment using Bayesian Adaptive Splines

Devin Francom[1], Bruno Sansó[2], Vera Bulaevskaya[3], Donald Lucas[4], Matthew Simpson[4]
[1]Los Alamos National Laboratory
[2]University of California Santa Cruz
[3]Climate Corporation
[4]Lawrence Livermore National Laboratory

December 7, 2018

Abstract

An atmospheric release of hazardous material, whether accidental or intentional, can be catastrophic for those in the path of the plume. Predicting the path of a plume based on characteristics of the release (location, amount and duration) and meteorological conditions is an active research area highly relevant for emergency and long-term response to these releases. As a result, researchers have developed particle dispersion simulators to provide plume path predictions that incorporate release characteristics and meteorological conditions. However, since release characteristics and meteorological conditions are often unknown, the inverse problem is of great interest, that is, based on all the observations of the plume so far, what can be inferred about the release characteristics? This is the question we seek to answer using plume observations from a controlled release at the Diablo Canyon Nuclear Power Plant in Central California. With access to a large number of evaluations of a computationally expensive particle dispersion simulator that includes continuous and categorical inputs and spatio-temporal output, building a fast statistical surrogate model (or emulator) presents many statistical challenges, but is an essential tool for inverse modeling and sensitivity analysis. We achieve accurate emulation using Bayesian adaptive splines to model weights on empirical orthogonal functions. We use this emulator as well as appropriately identifiable simulator discrepancy and observational error models to calibrate the simulator, thus finding a posterior distribution of characteristics of the release. Since the release was controlled, these characteristics are known, making it possible to compare our findings to the truth.

Keywords: model calibration, inverse problem, multivariate emulation, sensitivity analysis, categorical inputs, functional outputs, uncertainty quantification, atmospheric dispersion models

1 Introduction

Less than one year after California's Diablo Canyon Nuclear Power Plant became operational in 1985, a catastrophic nuclear accident occurred in Chernobyl, then part of the Soviet Union, resulting in large amounts of radioactive material being released into the atmosphere and proving fatal for many emergency responders and reactor staff. Radioactive plumes, which drifted over much of the western Soviet Union and Europe, necessitated the evacuation and long-term resettlement of many local people and have had lasting effects on public health (United Nations Scientific Committee on the Effects of Atomic Radiation, 2008).

Pacific Gas and Electric Company, which operates the Diablo Canyon Nuclear Power Plant, performed a series of experimental releases of sulfur hexafluoride, a benign gas, from the California plant soon after the 1986 Chernobyl accident. The purpose of these experiments was to gather concentration data at downwind observation sites after each release to be used for validation of established particle dispersion models in the presence of complex terrain (Thuillier, 1992). The inverse problem, to determine if the release characteristics can be inferred based on observations of the plume, was of less interest. However, the latest large-scale radioactive release that happened after an accident in Fukushima, Japan, has prompted renewed interest in the experimental release data from 1986 as a testbed for particle dispersion forward and inverse modeling (Lucas et al., 2017). Highly complex systems of this nature rarely allow for well controlled experiments, making the 1986 experimental release data a rare asset.

In the decades since the 1986 data have been analyzed, particle dispersion models and computational methods have evolved, and statistical calibration of computer models has become a well developed field, especially following the work of Kennedy and O'Hagan (2001). Computer models are used in a myriad of fields where a complex mathematical relationship (based on scientific understanding or hypothesis) between inputs and outputs is encoded for computer evaluation. Typically, these models are developed in areas where exhaustive experimentation is too expensive. Of course, substituting a model for an experiment has limitations. The field of uncertainty quantification has arisen in an effort to understand those limitations in practice.

Important tasks in uncertainty quantification often include understanding the effect of changing parameter settings (sensitivity analysis), determining the most probable parameter settings using experimental or observational data (calibration), and estimating the discrepancy between the model and reality. Hence, calibration solves the inverse problem mentioned above. When these methods for uncertainty quantification treat the computer model as a black box that takes input settings and returns an output, the methods are called "non-intrusive." While popular, non-intrusive methods are likely to require many evaluations of the computer model, which can be expensive in both time and computational resources. A way to decrease this expense is to build a fast statistical surrogate to the computer model, called an emulator, by using a relatively small number of evaluations of the computer model to fit the input/output relationship. A good emulator should quickly provide an accurate prediction of the computer model output for any parameter setting of interest. Again, using an emulator in place of the computer model has limitations. Those limitations are incorporated into the uncertainty quantification framework by propagating emulator uncertainty through to the other estimated uncertainties. Hence, it is important that, unless an emulator has negligible error, the error be accurately quantified.

Our interest lies in developing non-intrusive uncertainty quantification methods, specifically emulation, sensitivity analysis and inverse modeling methods, suitable for use with the 1986 atmospheric release observations and many evaluations of a state-of-the-art particle dispersion model. This is a difficult task due to the complexity of both the observed and simulated data. The experimental release observations form a spatio-temporal field that includes measurement error, background noise and missing values. Each of our many evaluations of the simulator has both continuous and categorical inputs and outputs a spatio-temporal field, resulting in massive amounts of simulated data. We note that, in general, uncertainty quantification has become an essential step in much of modern scientific exploration, and many fields are consistently increasing their computational capacity, making analysis of large amounts of simulated data a relevant topic.

The primary innovation in this work that can be extended to other large computer experi-

ments is our emulation methodology. We use Bayesian multivariate adaptive regression splines (BMARS), recently used for emulation with small numbers of model runs in Chakraborty et al. (2013), Stripling et al. (2013) and Francom et al. (2018). While BMARS can be used on its own for functional emulation (Francom et al., 2018), we opt for a different approach in this application. We use BMARS to model weights on spatio-temporal empirical orthogonal functions (EOFs) that are linearly combined to yield a spatio-temporal emulator. This has commonalities with the approach of Higdon et al. (2008), which emulates simulators with highly multivariate output by using Gaussian processes to model weights when linearly combining principle components. In our application, we also require an emulator that can handle categorical inputs. We develop a way for BMARS to incorporate categorical inputs that is true to its adaptive nature. The result is an emulator that (1) can flexibly and accurately model complicated functional response surfaces; (2) quantifies emulator uncertainty; (3) is scalable in the number of inputs, the number of model runs and the dimension of the functional response; and (4) has a closed form ANOVA decomposition that can be used for sensitivity analysis. This is all fairly easy to reproduce via the R package BASS (Francom, 2017; Francom and Sansó, in press).

While we present a general emulation framework that is well-suited to complex data like ours (mixed categorical and continuous inputs, functional response, many model runs), we note that this is not the only way to emulate this model. By far, the Gaussian process (GP) (Sacks et al., 1989) is the most commonly used model for emulation. However the GP is difficult to use in this case because we employ a large number of evaluations of the simulator, each with spatio-temporal output. Scalability is not a strength of the GP, though more scalable versions are discussed in Kaufman et al. (2011) and Gramacy and Apley (2015). These new versions, to our knowledge, will require some innovation to incorporate categorical variables, perhaps similar to the way that Qian et al. (2008) orZhang et al. (2018) incorporate them. Incorporating categorical variables into nonparametric regression methods is a non-trivial task, and is one of the innovations of this paper. Further, some approaches can handle large data at the expense of providing probabilistic uncertainty. Other approaches, like process convolutions (Higdon, 1998)

and predictive processes (Banerjee et al., 2008) suffer the curse of dimensionality. The purpose of this paper is not to compare emulation methods (though a limited comparison is included in the supplementary material), but is to demonstrate an effective emulation method that works and provides useful information for a complex case study.

Inverse modeling, or calibration, for this particular dataset presents its own set of unique challenges. In addition to emulation for simulators that are expensive to evaluate, other essential parts of calibration are building the simulator discrepancy model and the observational error model. We combine these models with a modularized Bayesian approach inspired by Liu et al. (2009) to ensure identifiability and to simplify computations. We use a BMARS model for the discrepancy. The scalability and flexibility of the combined framework could be valuable for other computer experiments with large amounts of observational and simulation data.

We introduce our data, methods and findings as follows. In Section 2, we introduce the experimental release data and our simulations. In Section 3, we detail the construction of our emulator and evaluate the fit. In Section 4, we describe our calibration framework, detailing simulator discrepancy and observational error models. We also demonstrate the inversion capability of the framework on synthetic data. In Section 5, we show the results of our calibration technique using the Diablo Canyon plume observations and discuss the accuracy. In Section 6, we summarize and detail future work.

2 Data

As is common in computer experiments, we have two sources of data: observations and simulations (evaluations of the simulator). In this section, we describe each of these.

2.1 Observations

The 1986 experimental release we analyze is one of eight releases performed on different days between August 31 and September 17. We focus on a release of 146 kilograms of sulfur hexafluoride

(SF_6) on September 4. Starting at 8:00AM local time, the SF_6 gas was released continuously for eight hours from a location at the base of the southmost containment unit at the Diablo Canyon Nuclear Power Plant. Air sampling was performed automatically at 150 sites in the surrounding area. At each site, air was pumped into a tedlar bag over the course of one hour, at which point the bag was sealed and air was pumped into a new bag. Sampling was done from the hours of 7:00AM to 7:00PM, yielding 12 measurements at each site. The quantity of interest, the concentration of SF_6, is reported as an hourly average for each site. Roughly 24% of the samples are missing for unknown reasons.

The first sample at each site (the 7-8am average) was taken prior to the beginning of the experimental release, thus measuring the background level of SF_6. Investigation of the background level shows moderate amounts near the plant switchyard, where SF_6 is used for electrical insulation. A map of the background level is shown in Figure 1, along with the time series of the period after the release for a few sites. The time series are noisy, show orders of magnitude in variation (even in the background level) and have missing values. The large background level at sites near the switchyard complicate the task of determining the controlled release characteristics based on our plume observations, since this fugitive release plays a confounding role. To minimize this confounding, for the 12 measurements at each site we subtract the site background level (the first measurement). Any negative values are truncated to zero. About 33% of the values were negative, though most were close to zero. Only 1.6% were less than -50. The largest value in the data is around 4×10^6, so a discrepancy of 50 is quite small. This is an effort to remove the background, but it makes the assumptions that the background level (1) is constant over time and (2) is not subject to large measurement error. The latter assumption is likely to be valid, but the first is difficult to justify. Still, in the absence of unconfounded temporal data to characterize the switchyard release, we proceed under these assumptions. For sites that are missing a background measurement (shown with slightly smaller dots in Figure 1), we spatially impute the background level from neighboring sites. There are no missing background readings that we believe would be significantly large (i.e., there are no missing values very near the switch-

yard). Because of the many orders of magnitude difference in observations, small background errors are unlikely to have much effect on our ability to identify release characteristics. Even the largest background measurements are orders of magnitude smaller than the largest post-release measurements later in the time series.

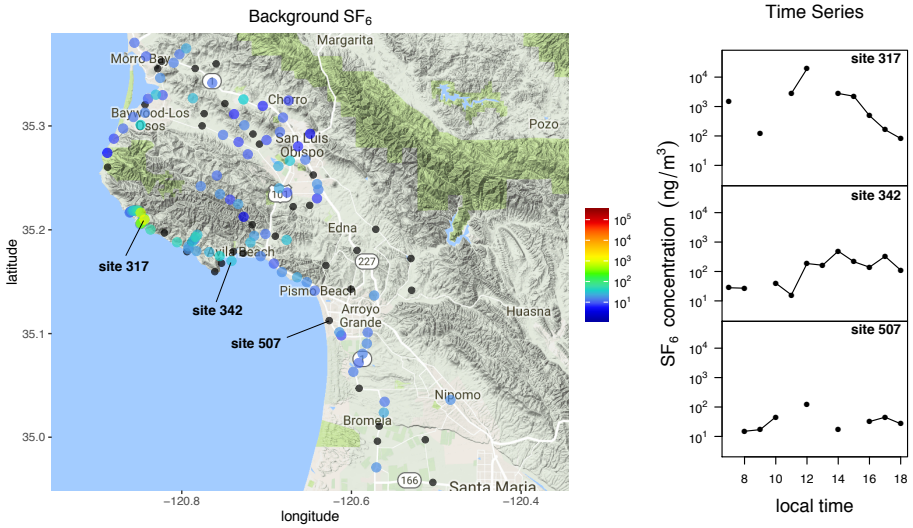


Figure 1: On the left, background SF_6 levels are shown (corresponding to 7:00 local time) for 137 of the 150 sites. Slightly smaller back dots are sites missing the 7:00 observation. On the right, observed time series for a few of the sites are shown. Lines connect adjacent observations in time, with missing values excluded.

2.2 Simulations

The simulator used in this work is a Lagrangian particle dispersion model called FLEXPART ("FLEXible PARTicle dispersion model") (Stohl et al., 2005). An in-depth discussion of the simulator and simulations can be found in Lucas et al. (2017), while we only introduce the simulations briefly here. The FLEXPART model requires six inputs detailing characteristics of the release (latitude, longitude, altitude, start time, duration, and amount) as well as a wind field. We use the Weather Research and Forecasting Model (WRF) to generate wind fields while varying only a few of WRF's many inputs. The inputs we vary are pre-release initialization time (9 or 15 hours), planetary boundary layer physics model used (YSU, MYJ TKE, or MYNN TKE), land surface model used (thermal diffusion, Noah, or RUC), FDDA nudging amount

(none, low, or high), and type of reanalysis data used (NARR, ECMWF, or CFSR). We use a series of five nested domains for evaluating WRF, so that the innermost domain resolves winds to 300 meters. More about these parameters and nesting can be found in Skamarock et al. (2008). A combination of the five categorical variables yields a wind field, which, along with a setting of the six continuous variables detailing the release characteristics, yields a simulated plume.

We obtain an ensemble of 18000 evaluations of FLEXPART with the 11-dimensional inputs sampled from a Latin Hypercube using the ranges in Table 1 for the six continuous parameters. We note that 18000 model evaluations is unusually large in the computer experiment literature, though it is becoming more common to have large numbers of model runs (Gramacy and Apley, 2015; Kaufman et al., 2011). Our computational budget allowed for this large number of evaluations, though each is still expensive to obtain. As discussed above, this large number of simulations makes emulation difficult, since traditional emulation techniques do not scale well.

The output from one evaluation of the simulator is spatio-temporal on a grid of 400×400 spatial locations and 34 time points, as shown in Figure 2. The 34 time points are 30-minute averages starting at 6:00 local time, thus extending to 23:00. We use only a subset of the 160,000 spatial locations when building the emulator, though the methods we present are scalable to moderately large spatial grids. Specifically, we use the 137 spatial locations that correspond to the sites shown in Figure 1, which are sites where we collect measurements. We exclude 13 of the 150 measurement sites from the analysis because they fall outside the region considered in the simulations, and are far enough away from the release that they are likely to only measure background levels.

Both simulation and observation data are transformed to be on the $\log_{10}$ scale after adding a constant to all values. The log scale makes modeling the many orders of magnitude difference in concentrations easier. Adding a constant is essential because the simulations have numerous zeros and very small values. If the constant is small relative to the range of the data, say 10^{-5}, we turn each zero into a -5. Since our largest values in the data are around 10^6, or 6 on the $\log_{10}$ scale, emulation in this case would devote significant effort to understanding variation

Table 1: FLEXPART continuous parameter ranges with true release values

parameter	lower bound	upper bound	true value
latitude	35.1977	35.2250	35.2111
longitude	-120.8708	-120.8384	-120.8543
altitude (meters)	1	10	2
start time	7:00	9:00	8:00
duration (hours)	6	10	8
amount (kg)	10	1000	146.016

in extremely small concentrations. Adding 50, determined in consultation with field experts, minimizes the amount of weight we give to small (in this case unimportant) concentration values that are indistinguishable from background values in practice.

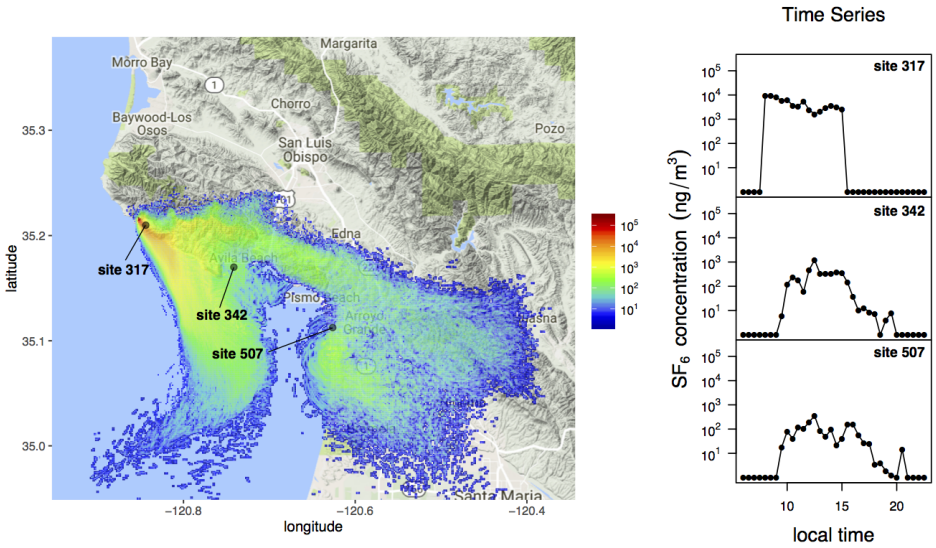


Figure 2: We show the simulated SF_6 plume for one of the 18000 simulations on the left (corresponding to 9:30 local time). On the right, we show simulation time series for a few of the sites.

3 Emulator

A usable emulator in this case needs to be able to produce a reasonably close approximation to the computer model for any possible combination of the 11 inputs. That is, given the characteristics of the release and the wind field characteristics, it should produce the spatio-temporal plume. Obtaining an estimate of emulation error is also a necessary task in order to allow us to propagate

the error into calibration uncertainty. If this error is disregarded, we leave open the possibility of being too confident in our estimates of release and wind characteristics that could have generated the calibration data. This case also requires an emulator that can use the large number of model runs available.

Our model runs provide us with 18000 plumes in space and time. We use these plumes to obtain empirical orthogonal functions (EOFs) in space and time jointly, and we model the weights in this EOF decomposition using adaptive splines. In cases where a large number of model are runs available, like this one, a very reasonable question is whether an emulator is needed at all. If we can run the simulator this many times, why do we need an emulator when solving the inverse problem? Large ensembles of expensive computer model runs are becoming more common as supercomputers become larger. This is only because the ensembles can be run in parallel. A sequential algorithm that chooses where to do the next run (like the Markov chain Monte Carlo approach to solving the inverse problem) would require serial, or nearly serial, model evaluations, which can be extremely time consuming even if the simulator only takes a few minutes.

One might also consider the prospect of obtaining so many model runs that some nearest neighbor or interpolation approach could be used as an emulator with negligible error. It should be noted that in 11 dimensional space, 18000 evaluations is still quite a small number. If we broke each of the 11 dimensions into 3 bins and put a point in each bin, we would use $3^{11} = 177147$ points, roughly 10 times the number that we have, and 3 bins in each dimension is quite a coarse grid. We also note that, as a first step in our analysis, we tried to find the collection of model runs that best matched the observations (or a left out simulation) based on minimizing mean square error and maximizing correlation. We found that that objective function can be quite noisy, having optimum far from true values. Our Bayesian model calibration approach that utilizes the emulator provides more refined estimates. Further, those estimates are coupled with probabilistic uncertainty, which is important to scientists. The emulator is also a useful tool on its own, since it can be used for sensitivity analysis.

3.1 Empirical Orthogonal Functions

Let $y^c(s,t,\mathbf{x})$ denote computer model output at spatial location s, time t and 11-dimensional input setting $\mathbf{x}$. The model output is on a grid of n_s spatial locations and n_t times. We define the vector of model output for input setting $\mathbf{x}_j$ as

$$\mathbf{y}^c(\mathbf{x}_j) := (y^c(s_1,t_1,\mathbf{x}_j), y^c(s_2,t_1,\mathbf{x}_j), \ldots, y^c(s_{n_s},t_1,\mathbf{x}_j), y^c(s_1,t_2,\mathbf{x}_j), \ldots, y^c(s_{n_s},t_{n_t},\mathbf{x}_j))' \quad (1)$$

and define the matrix of model run output $\mathbf{Y}^c := [\mathbf{y}^c(\mathbf{x}_1), \ldots, \mathbf{y}^c(\mathbf{x}_{n_x})]$ for the n_x model runs, which has dimensions $n_s n_t \times n_x$. We obtain discretized EOFs using the singular value decomposition, yielding $\mathbf{Y}^c = \mathbf{U}\mathbf{D}\mathbf{V}'$. The matrix $\mathbf{U}$ is the $n_s n_t \times n_s n_t$ matrix that has EOFs as columns and $\mathbf{D}\mathbf{V}'$ is the $n_s n_t \times n_x$ matrix of weights. To reduce the dimension in the problem, we use only the first k EOFs, where $k < n_s n_t$, resulting in the truncated decomposition $\hat{\mathbf{Y}}^c = \hat{\mathbf{U}}\hat{\mathbf{D}}\hat{\mathbf{V}}'$ where $\hat{\mathbf{U}}$ has dimension $n_s n_t \times k$ and $\hat{\mathbf{D}}\hat{\mathbf{V}}'$ has dimension $k \times n_x$. For modeling purposes, we write the non-truncated EOF decomposition as $y^c(s,t,\mathbf{x}_j) = \sum_{i=1}^{n_s n_t} K_i(s,t) w_{ij}$ where $K_i(s,t)$ is the i^{th} EOF at spatial location s and time t and w_{ij} is the corresponding weight for $\mathbf{x}_j$, specifically $(\mathbf{D}\mathbf{V}')_{ij}$.

We specify our emulator as the truncated decomposition

$$y^c(s,t,\mathbf{x}) = \sum_{i=1}^{k} K_i(s,t) w_i(\mathbf{x}) + u(s,t) \quad (2)$$

where $u(s,t)$ is the truncation error. We replace w_{ij} with $w_i(\mathbf{x})$ in order to allow us to predict computer model output for $\mathbf{x}$ not in our training sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n_x}\}$. We model the weight functions using adaptive splines with

$$w_i(\mathbf{x}) = \eta_i(\mathbf{x}) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma_i^2) \quad (3)$$

using the values of $\{w_{i1}, \ldots, w_{in_x}\}$ to train $w_i(\cdot)$. We discuss the form of the function $\eta_i(\cdot)$ in the next section. We assume that the weights on the EOFs are independent *a priori*.

Regarding the truncation, we need k, the number of EOFs used, to be sufficiently large to yield a suitable approximation, but small enough to be computationally feasible. Assuming a parametric distribution for the truncation error, $u(s,t)$, for which it is independent and identically distributed for all s and t is likely to improperly characterize the emulation uncertainty because of the large variation in plume characteristics in space and time. Instead we assume truncation error for a particular (s,t) combination comes from the distribution of $n_x = 18000$ truncation errors we have seen already,

$$u(s,t) \sim Unif\left\{ y^c(s,t,\mathbf{x}_j) - \sum_{i=1}^{k} K_i(s,t) w_{ij} \right\}_{j=1}^{n_x}. \tag{4}$$

Using a reduced number of EOFs is what allows us to accomplish dimension reduction. Truncating the number of EOFs results in a (known) deviation between the true data and the reduced dimension approximation. For each input combination in our training data, we know the deviation between the full spatio-temporal field and the reduced dimension prediction of the spatio-temporal field. Hence, for each spatio-temporal location, we have a distribution of n_x deviations. The discrete uniform distribution in Equation 4 samples from these n_x deviations. Since the EOFs are fixed, there is little interest in the truncation error after we have chosen k, other than to make sure it is propagated during calibration. Hence, we see no value in trying to estimate a parametric distribution. The large value of n_x makes our discrete distribution for the truncation error accurate without making any limiting assumptions about distribution tails and symmetry. Further, since this distribution has no unknown parameters and because the weights $w_i(\mathbf{x})$ and $w_j(\mathbf{x})$ have no *a priori* correlation, the weights will also have no *a posteriori* correlation. This means that the adaptive spline models for $w_i(\mathbf{x})$ and $w_j(\mathbf{x})$ can be fit completely in parallel, which is a significant computational benefit.

3.2 Adaptive Splines

BMARS models are a Bayesian version of the more traditional MARS models introduced in Friedman (1991b). These models can be a powerful tool for emulation for a number of reasons (Denison et al., 1998; Chakraborty et al., 2013; Francom et al., 2018): (1) they are adaptive because knots and variables are only included in basis functions if they are necessary; (2) they perform implicit variable selection, as basis functions only include variables that are useful; (3) they scale well, especially when compared to Gaussian process models; (4) they can flexibly pick up localized signal when the data suggest such signal exists; (5) they yield analytical Sobol' sensitivity indices (see Section 2 of the supplementary material, Section 3.4, and Francom et al. (2018)); and (5) they can be used to emulate simulators with functional response.

The model for the adaptive spline mean function $\eta_i(\mathbf{x})$, used in Equation 3, is a linear combination of tensor product basis functions. We drop the index as we describe this model, since it is fit independently for $i = 1, \ldots, k$. Thus, $\eta(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m B_m(\mathbf{x})$ where the basis function $B_m(\mathbf{x})$ is of the form

$$B_m(\mathbf{x}) = \begin{cases} \prod_{j=1}^{J_m} g_{jm}[s_{jm}(x_{v_{jm}} - t_{jm})]_+ & \text{if } J_m > 0 \\ 1 & \text{if } J_m = 0. \end{cases} \tag{5}$$

When $J_m > 0$, the basis function is a tensor product of polynomial splines where t_{jm} is a knot, v_{jm} indexes a variable, and $s_{jm} \in \{-1, 1\}$. The function $[\cdot]_+$ is defined as $\max(0, \cdot)$. The constant g_{jm} scales the basis function to have maximum of one, which helps with computational stability. Without loss of generality, if $x_{v_{jm}} \in [0, 1]$, then $g_{jm} = [(s_{jm} + 1)/2 - s_{jm} t_{jm}]^{-1}$. A basis function has J_m elements in the tensor product where each is required to involve a different variable. Hence, J_m is the degree of interaction for basis function m. The piecewise structure of the basis function will become important in the next section, where we will discuss the case when $J_m = 0$.

The unknowns associated with $\eta(\mathbf{x})$ are the number of basis functions, M, the basis function

weights $\mathbf{a}$, and the basis function parameters $\boldsymbol{\psi} := \{J_m, (t_{jm}, v_{jm}, s_{jm})_{j=1}^{J_m}\}_{m=1}^{M}$. The value of σ^2 from Equation 3 is also unknown (again, we drop the index for notational simplicity). We assign priors to all of these parameters following the specifications of Francom et al. (2018). To keep the number of basis functions small, we use $M|\lambda \sim Poisson(\lambda)$ with $\lambda \sim Gamma(1, 1 \times 10^5)$. We also regularize the basis coefficients by using a Zellner-Siow Cauchy prior (Zellner and Siow, 1980; Liang et al., 2008) such that $\mathbf{a}|\sigma^2, \tau, \mathbf{B} \sim N(\mathbf{0}, \sigma^2(\mathbf{B}'\mathbf{B})^{-1}/\tau)$ with $p(\sigma^2) \propto 1/\sigma^2$ and $\tau \sim Gamma(1/2, 2/n_x)$. Here, $\mathbf{B}$ is the $n_x \times (M+1)$ matrix of basis functions (including an intercept) conditional on the number of basis functions and the basis function parameters $\boldsymbol{\psi}$. These basis function parameters are given a discrete uniform prior (i.e., each possible combination is given equal weight) with the constraint that each resulting basis function have at least 20 non-zero points to help prevent edge effects. This generally follows the model formulation of Francom et al. (2018), excluding the functional response.

The approach of Francom et al. (2018) models the functional output by including the functional variable in the BMARS basis functions via tensor products. When the functional response has multiple variables and is complex in shape (like the plumes in this paper), the approach of using BMARS to model weights in another basis expansion allows us to use fewer basis functions in each BMARS model. This is because the EOFs, which are derived from the training data, tend to describe complex shapes of a particular dataset in an efficient manner. Even though we use a large number of BMARS models (one for each EOF weight), each can be fit in parallel, meaning that the emulator can be built quickly. When fitting a BMARS model, conditional on the set of basis functions in any given RJMCMC step, we have a linear system that we solve using a Cholesky Decomposition, which requires number of operations cubic in the number of basis functions. The Francom et al. (2018) approach might, for example, use 1000 basis functions to capture the complicated relationships between the input and complex output, as well as the relationships within the functional output. The EOF approach, on the other hand, might use 100 basis functions for each of the 100 models (one for each EOF). That would result in an order of magnitude fewer computations for the EOF approach. Hence, for many situations where the

output is complicated, the EOF approach may make more sense.

3.3 Categorical Predictors

As we have discussed, our emulator needs to be able to handle categorical inputs. This requires an extension of the traditional adaptive splines models that follows an approach similar to Friedman (1991a), but modified to fit into our Bayesian framework. Specifically, if the inputs to the simulator are $(\mathbf{x}, \mathbf{z})$ where $\mathbf{z}$ is a vector of categorical variables, we define the portion of the basis function due to the categorical variables as

$$B_m(\mathbf{z}) = \begin{cases} \prod_{l=1}^{L_m} 1(z_{u_{lm}} \in D_{lm}) & \text{if } L_m > 0 \\ 1 & \text{if } L_m = 0. \end{cases} \tag{6}$$

where u_{lm} indexes a categorical variable, D_{lm} is a proper subset of the categories of variable u_{lm}, and L_m is the degree of interaction of categorical variables. This approach to handling categorical variables is somewhat similar to Storlie et al. (2015) and Ma et al. (2015). We combine this portion of the basis function with Equation 5 so that the m^{th} basis function is $B_m(\mathbf{x}, \mathbf{z}) = B_m(\mathbf{x})B_m(\mathbf{z})$. The purpose of allowing for $L_m = 0$ or $J_m = 0$ is to allow for basis functions that involve only the continuous predictors, or only the categorical predictors, respectively. In our case study, we permit up to third-order interactions of each variable type ($J_m = 3$ and $L_m = 3$) maximally resulting in six-way interactions if the data dictates that such interactions are useful. Allowing D_{lm} to be a subset of categories rather than a single category is useful for cases when two or more categories exhibit similar behavior, as it allows us to use fewer basis functions to describe the behavior. We again use a discrete uniform prior for the categorical basis function parameters $\boldsymbol{\phi} := \{L_m, (u_{lm}, D_{lm})_{j=1}^{L_m}\}_{m=1}^{M}$, and we alter the constraint discussed above to apply to the new set of basis functions that include both categorical and continuous variables. That is, each resulting basis function must have at least 20 non-zero points, but otherwise, each combination of parameters conditional on interaction degree has equal prior

probability.

Because the resulting posterior is transdimensional (since M is unknown), we use reversible jump Markov chain Monte Carlo (Green, 1995) to obtain samples of the posterior model space. We use the steps detailed in Denison et al. (1998) with the improvements of Nott et al. (2005) to add a basis function, delete a basis function, or make a small change to a basis function. These steps are described in detail in the supplementary material.

The full formulation of the emulator is then given by

$$y^c(s,t,\mathbf{x},\mathbf{z}) = \sum_{i=1}^{k} K_i(s,t) w_i(\mathbf{x},\mathbf{z}) + u(s,t) \tag{7}$$

$$w_i(\mathbf{x},\mathbf{z}) = \eta_i(\mathbf{x},\mathbf{z}) + \epsilon_i \tag{8}$$

$$\eta_i(\mathbf{x},\mathbf{z}) = a_{0i} + \sum_{m_i=1}^{M_i} a_{m_i} B_{m_i}(\mathbf{x},\mathbf{z}) \tag{9}$$

with the priors for the parameters in $B_{m_i}(\cdot,\cdot)$, which are $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$, and the prior for ϵ_i as specified above.

3.4 Sensitivity Analysis

A major benefit of this emulator formulation is the ability to infer the importance of particular variables or interactions between variables. Specifically, we can partition the variation in the emulator in the same way that variance is partitioned in a linear model (through ANOVA). This variance decomposition is achieved through the Sobol' decomposition (Sobol', 1990, 2001; Saltelli et al., 2008), and is used to perform Global Sensitivity Analysis in the UQ literature. The Sobol' decomposition is typically approximated (via Monte Carlo methods) for non-linear models because of the need to calculate high-dimensional integrals. Francom et al. (2018) showed the analytical solutions to these integrals for BMARS models with continuous predictors. We extend that work to show the analytical solutions for BMARS models with categorical predictors. For maximum utility, we also show how to obtain the solutions when modeling in EOF space. These developments are shown in detail in the Supplementary Material. They represent the ability to

see, for each spatial location and time point, the partitioned variance of the emulator. This is one of the major advantages of our proposed model.

3.5 Performance

We demonstrate the performance of the emulator by comparing emulator and simulator output at input settings not used to fit the surrogate or to build the EOFs. We use the R package BASS (Francom, 2017) to fit the BMARS models. Figure 3 shows the emulator prediction performance at 15 of the 137 spatial locations for two different holdout simulator runs. In addition to the emulator mean, posterior predictive uncertainty is shown. The prediction uncertainty varies with space and time because (1) the homoscedastic BMARS error from Equation 3 is multiplied by the corresponding spatio-temporal EOF in Equation 2, thus producing spatio-temporal noise and (2) the truncation error varies in space and time. The models corresponding to the most important EOFs (the EOFs corresponding to larger singular values, which are the diagonal entries of $\mathbf{D}$) use around 200 BMARS basis functions to fit the data, while the least important EOFs, which correspond to higher frequencies, use around 50 basis functions.

We note that RJMCMC algorithms typically mix poorly, and that gauging RJMCMC convergence can be difficult. Monitoring convergence for 100 chains is particularly cumbersome. In practice, it may be that the best we can say for high-dimensional models like this is that we have converged to a high-probability region of the posterior. While we could use tempering (Francom et al., 2018) to ensure that our RJMCMC sampler is not stuck in a local mode of the posterior, efficient selection of a temperature ladder is difficult. We find that running multiple chains and checking to see that trace plots of the Sobol' indices for the chains converge to the same values can be an indication that we are sampling from the highest-probability region. We opt to burn in for 190000 iterations, and then keep every 10th iteration as a sample from the posterior until we get 1000 samples. On a personal computer, the 200000 iterations take roughly 40 minutes for the more important EOF models (which use more BMARS basis functions) and around 20 minutes for the models that use fewer basis functions. These models can be fit in parallel.

The holdout predictions in Figure 3 demonstrate reasonable emulator performance, especially considering the complexity of this simulator. Predictions of a number of other randomly chosen holdout simulations also show good performance. By using a large number of EOFs in our emulator, we are able to capture subtle variation in the shape of the time series for different input settings. For instance, the difference in the shapes of the two time series for the Avila Beach location (see Figure 3 (a) and (b)) could be attributed to "random" variation if a less accurate emulator was used. Instead, we can attribute the difference to parameter variation. (Because the simulator is deterministic, there is technically no "random" variation, though there may be numerical variation that is difficult to attribute to any parameter.) The particular parameter variation that is important is demonstrated through the functional Sobol' decomposition in Figure 4. This is powerful because it shows exactly what is contributing to the variation in the emulator, demonstrating, for instance, that the differences between our plotted predictions at the Avila Beach site are likely due to a change in the reanalysis scheme and amount released.

Numerous other important characteristics can be discovered from the functional sensitivity analysis in Figure 4. For instance, main effects and interactions including variables 3, 8 and 9 do not show up in the 11 most important effects, indicating that the model is less sensitive to those parameters. In fact, variable 3 is virtually unused in the emulator, which we discuss later. It is also interesting to note that the location of the release explains much of the variation in the locations close (and downwind) to the release, but very little of the variation for the distant locations. The empty plots for Morro Bay and San Luis Obisbo indicate that there was no variation there (the plume never reached those locations), while the small variation in the Edna plot shows that there were some plumes that made it there. The duration (variable 5) explains variation in the end of the time series while the reanalysis scheme explains variation at the beginning. The start time (variable 4) has little impact except at the early parts of the time series for close locations. Interestingly, the WRF initialization time (variable 7) had a significant impact on the "midway up See Canyon" site, a location with considerably complex terrain. At all sites, and for much of the time series, the release amount (variable 6) is a significant

driver. Also, there is very little variation coming from four-way interactions and above. All of there characteristics can be useful for domain scientists and demonstrate the significant benefit performing sensitivity analysis using this emulation technique.

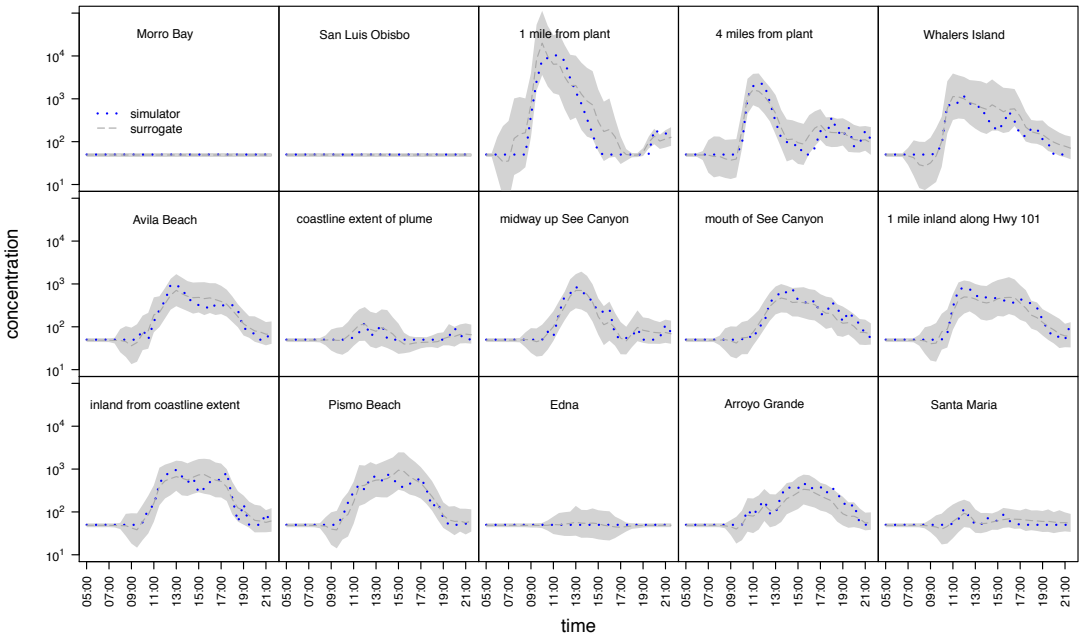
Our efforts to build an emulator for a single location (using the 34 time points) resulted in poor emulation compared to those that utilize the spatio-temporal plume information. We also achieved better emulation performance using spatio-temporal EOFs than we did using EOFs that were separable in space and time. While separable EOFs have the benefit of easier interpretation, we have little interest in interpretation, and separability ends up being a poor assumption for this plume model. Interpretation of the EOFs is unimportant because we can perform the functional sensitivity analysis to understand exactly what is going on in the model, rather than trying to understand what is going on in terms of EOFs.

3.6 Other Emulators

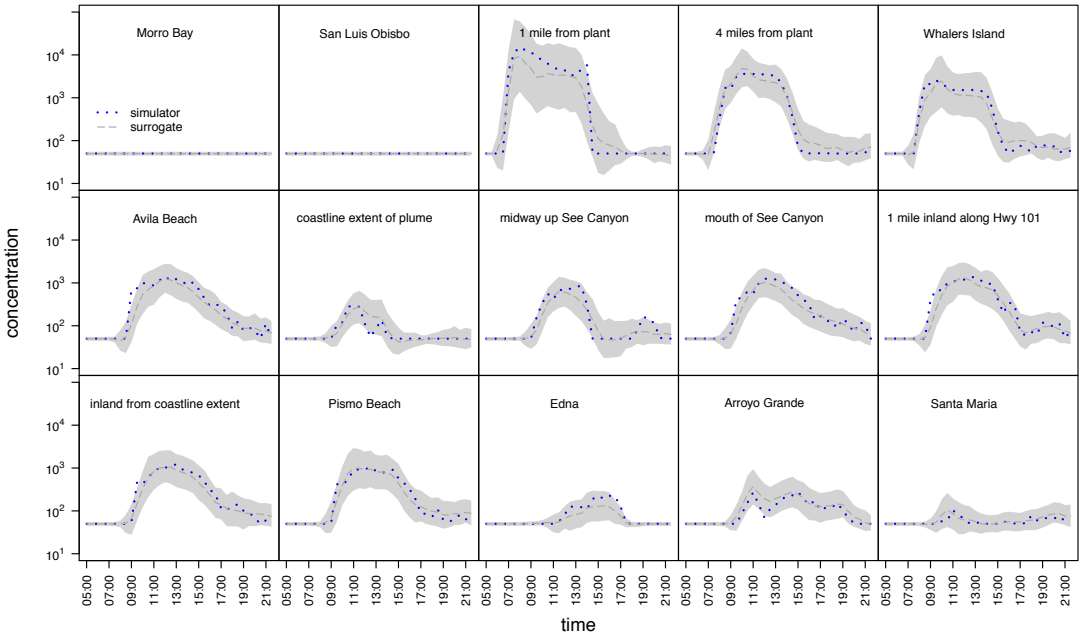
Other emulators could have been used here. In particular, we could have used a different nonlinear regression model to emulate the weights on the EOFs. The class of emulators is limited by the fact that we care about quantifying prediction uncertainty and we have both categorical and continuous inputs. For the purposes of sensitivity analysis, BMARS is very well suited to this problem, but in general, one has to make a choice between emulators. Here we provide a brief discussion of choosing between emulators, with the appropriate justification for our BMARS emulator. We compare the BMARS emulator to two other possibilities: (1) treed Gaussian processes (TGP) and (2) Bayesian additive regression trees (BART).

TGP is a powerful version of the GP that partitions the input space and fits independent GPs to each partition. The partitions and the GP parameters are learned jointly with Bayesian methods. While categorical variables can be difficult to incorporate into a GP model, TGP very naturally incorporates them into the tree structure. BART uses a sum of trees approach, where trees can split on categorical and continuous parameters.

We compare the three models based on their prediction of the first EOF weight, which is the



(a) Holdout Sample 1



(b) Holdout Sample 2

Figure 3: Demonstration of emulator fit at 15 locations. The top plots (a) show the time series simulator output (dotted lines) for a particular set of inputs not used to train the emulator. The bottom plots show a different input setting, to demonstrate variation in the model output shape. Emulator posterior predictive means (dashed lines) and pointwise 95% probability intervals (shaded region).

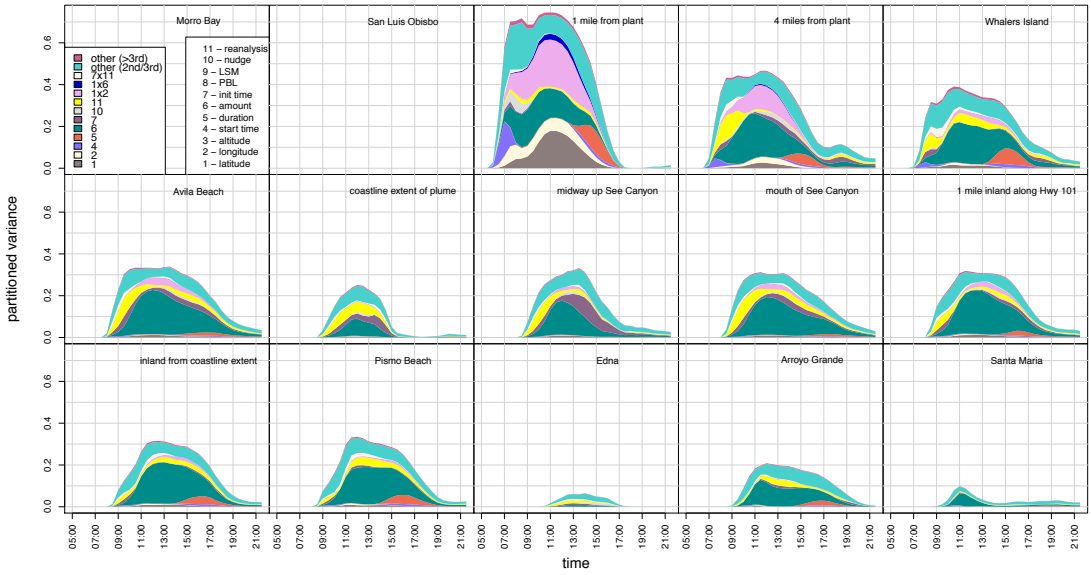


Figure 4: Functional sensitivity analysis of the EOF adaptive spline emulator. For each location, the emulator variance as a function of time is partitioned based on the 11 most important main effects and interactions. For example, at the "Avila Beach" site, at 9:00, the largest effect on the variation in the emulator is the reanalysis scheme.

most important EOF weight. In the supplement, we also compare the models for the 50th EOF weight (to represent something with a lower signal to noise ratio) as well as another example function.

4 Calibration Model

Our strategy for calibration differs from that of Kennedy and O'Hagan (2001) in a few important ways. Rather than the GP emulator, we use BMARS. We also opt to fit the emulator before calibrating, independent of the observational data. While we do this primarily for computational reasons, it marks a general difference in strategy. A similar approach was used in Gramacy et al. (2015), with the justification that with a large number of simulations the observational data is not going to significantly influence the emulator (i.e., there will be no empirical difference). Others justify this strategy on philosophical grounds, because the emulator is meant only to replicate the computer model and thus should not be influenced by observational data (Liu

et al., 2009).

Let $y^F(s,t)$ denote the log concentration data gathered from the sensor at location s and time t. Let $\zeta(s,t)$ denote the true log concentration at the same location and time. Then we set up the calibration model as follows:

$$y^F(s,t) = \zeta(s,t) + \nu, \quad \nu \sim N(0, \sigma_F^2) \tag{10}$$

$$\zeta(s,t) = y^c(s,t,\boldsymbol{\theta}_x,\boldsymbol{\theta}_z) + \delta(s,t) \tag{11}$$

where ν is the observation error, σ_F^2 is the observation error variance and $y^c(s,t,\boldsymbol{\theta}_x,\boldsymbol{\theta}_z)$ denotes the estimated computer model output at location s, time t, continuous input setting $\boldsymbol{\theta}_x$ and categorical input setting $\boldsymbol{\theta}_z$. The $\delta(s,t)$ term denotes the systematic model discrepancy. Hence, $\boldsymbol{\theta} = (\boldsymbol{\theta}_x, \boldsymbol{\theta}_z)$ is the unknown setting of the simulator that best matches reality when jointly considered with simulator discrepancy and observational error. While the observations are hourly averages, the emulator gives 30-minute averages. Thus, we average 30-minute averages in the emulator output to match the observation time scale. We also exclude emulator and discrepancy values at (s,t) where the corresponding $y^F(s,t)$ is missing.

A potential problem with the introduced modeling framework is that the observations can be produced in a number of different ways. For instance, we might get good prediction if we get as close as we can to the observations by only altering $\boldsymbol{\theta}$, and then consider $\delta(\cdot)$ to be the leftover spatio-temporal structure in combination with ν, the observational error. However, we could achieve equally good prediction by fixing $\boldsymbol{\theta}$ at a particular value and only altering $\delta(\cdot)$. These two examples represent the extremes in overfitting, but we may attain equally misleading combinations of these (see Liu et al. (2009) for examples). Hence, we need restrictions in order to make all the terms identifiable. The most satisfying restrictions we can introduce are in the form of an informative prior for the shape of the discrepancy, rather than an informative prior for the parameters $\boldsymbol{\theta}$ (Liu et al., 2009). While there may be some applications where the shape of the discrepancy is somewhat understood, we do not have an informative prior for our discrepancy

model.

Following Liu et al. (2009), we make the quantities of interest identifiable by modularizing the analysis further. Particularly, we fix the shape of the model discrepancy before trying to infer $\boldsymbol{\theta}$. We get an estimate of the model discrepancy by estimating the computer model output at the prior mean parameter settings (Bayarri et al., 2007) and subtracting that from the observations. We then fit a BMARS model to that discrepancy as a function of s and t, effectively smoothing it out. While we fix this discrepancy shape, we allow for its influence to vary by multiplying the discrepancy by a scale factor, γ. Our prior for γ is uniform between zero and two. If γ is near zero, the influence of the discrepancy is minimal. If it is near two, then the discrepancy plays a larger role. We limit the upper bound to two because anything larger would give the discrepancy similar magnitude to the simulator output, which we hope is not the case. While we introduce a scale parameter to the discrepancy, we do not introduce an intercept parameter because doing so would likely confound our ability to learn one of the simulator parameters (release amount), which explains much of the variation in magnitude of the concentration. Without *a priori* knowledge of systematic magnitude discrepancy, we refrain from including an intercept (or a multiplicative discrepancy term for $y^c(\cdot)$, included in Kennedy and O'Hagan (2001)). Thus, we alter Equation 11 to be

$$\zeta(s,t) = y^c(s,t,\boldsymbol{\theta}_x,\boldsymbol{\theta}_z) + \gamma\delta(s,t).$$

While the functional forms of $y^c(\cdot)$ and $\delta(\cdot)$ are fixed in advance, we incorporate their uncertainties in calibration by sampling their posterior predictive distributions, rather than using their mean functions.

We note that different structures of the measurement error variance σ_F^2 could be modeled. However, the mean of the corresponding Gaussian distribution is a complicated function. Typically, a complicated variance function would be difficult to identify, and possibly unnecessary conditional on the complicated mean function. Our collaborators had the opinion that, given the tedlar bags had the same kind of pumps and were taken to the same lab for measurement,

there should not be a site-specific measurement error.

Under this calibration framework, our likelihood is $\mathbf{y}^F|\boldsymbol{\theta}_x, \boldsymbol{\theta}_z, \gamma, \sigma^2 \sim N(\mathbf{y}^c(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z)+\gamma\boldsymbol{\delta}, \sigma_F^2\mathbf{I})$ and our posterior is

$$\pi(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z, \sigma_F^2, \gamma|\mathbf{y}^F) \propto N(\mathbf{y}^F|\mathbf{y}^c(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z) + \gamma\boldsymbol{\delta}, \sigma_F^2\mathbf{I})IG(\sigma_F^2|a, b)1(\boldsymbol{\theta}_x \in D_x)1(\boldsymbol{\theta}_z \in D_z)1(\gamma \in [0, 2]) \tag{12}$$

where D_x is the hypercube based on the prior ranges identified in Table 1, D_z allows for all the discrete parameter combinations, and $\mathbf{y}^c(\cdot)$ is defined in Equation 1. We obtain samples from the posterior by using Markov chain Monte Carlo (MCMC) methods (Gelman et al., 2013). We sample σ_F^2 and γ from their Inverse Gamma and Truncated Normal full conditionals, respectively. We use the Metropolis-Hastings algorithm to sample the six continuous parameters $\boldsymbol{\theta}_x$ from their joint full conditional. We sample the five categorical parameters $\boldsymbol{\theta}_z$ jointly from their discrete full conditional. Specifically, there are 162 combinations of the categorical parameters. We take a sample from the emulator posterior predictive distribution for each of the 162 combinations and evaluate the posterior up to a constant. This is reweighted to produce a posterior (full conditional) probability for each setting of the categorical parameters. We then sample one of the 162 combinations according to that probability distribution. Specifically, we sample according to the following scheme:

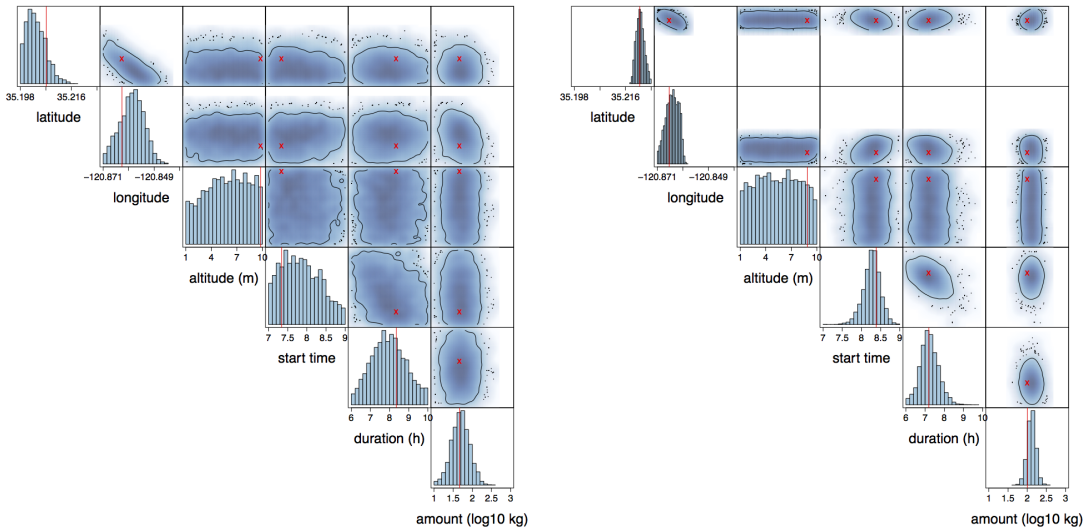
$$\begin{aligned}
[\boldsymbol{\theta}_x|\boldsymbol{\theta}_z, \gamma, \sigma^2] &\propto N(\mathbf{y}^F|\mathbf{y}^c(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z) + \gamma\boldsymbol{\delta}, \sigma_F^2\mathbf{I})1(\boldsymbol{\theta}_x \in D_x) \\
Pr(\boldsymbol{\theta}_z = \mathbf{z}|\boldsymbol{\theta}_x, \gamma, \sigma^2) &\propto N(\mathbf{y}^F|\mathbf{y}^c(\boldsymbol{\theta}_x, \mathbf{z}) + \gamma\boldsymbol{\delta}, \sigma_F^2\mathbf{I})1(\mathbf{z} \in D_z) \\
[\sigma_F^2|\boldsymbol{\theta}_x, \boldsymbol{\theta}_z, \gamma] &\sim IG(\sigma_F^2|a + N_F/2, b + (\mathbf{y}^F - \mathbf{y}^c(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z) - \gamma\boldsymbol{\delta})'(\mathbf{y}^F - \mathbf{y}^c(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z) - \gamma\boldsymbol{\delta})/2) \\
[\gamma|\boldsymbol{\theta}_x, \boldsymbol{\theta}_z, \sigma_F^2] &\sim TN(\boldsymbol{\delta}'(\mathbf{y}^F - \mathbf{y}^c(\boldsymbol{\theta}_x, \boldsymbol{\theta}_z))/\boldsymbol{\delta}'\boldsymbol{\delta}, \sigma^2/\boldsymbol{\delta}'\boldsymbol{\delta}, 0, 2).
\end{aligned}$$

The computational bottleneck to this algorithm is sampling the emulator posterior predictive distribution, as it requires building the BMARS basis functions for all EOF models. A single

sample takes 0.2 seconds when the tasks for the 100 models are split between four cores. To obtain 162 samples, one for each categorical combination used to form the distribution $Pr(\boldsymbol{\theta}_z = \mathbf{z}|\boldsymbol{\theta}_x, \gamma, \sigma^2)$, a naive approach would require more than 30 seconds for each MCMC iteration. We overcome the bottleneck in sampling the posterior predictive 162 times by building all of the possible categorical basis functions in advance. For each EOF model and each emulator posterior sample, we obtain the basis functions used in each of the 162 predictive combinations. The resulting basis functions, which are combinations of ones and zeros, are multiplied (pointwise) by the portions of the basis functions from the continuous predictors. Specifically, from Equations 7-9, we minimize the calculation of $B_{m_i}(\mathbf{x}, \mathbf{z}) = B_{m_i}(\mathbf{x})B_{m_i}(\mathbf{z})$ by precomputing and storing in memory all the $B_{m_i}(\mathbf{z})$ that occur in the emulator posterior samples. This allows us to obtain the 162 posterior predictive samples in less than 0.1 seconds on four cores, in contrast to the naive approach that takes more than 30 seconds. Though the categorical basis functions may seem like they would have a large memory footprint, they are combinations of ones and zeros and thus can be stored efficiently in memory.

4.1 Synthetic Calibration

To test our methods, we can calibrate to data where we know the truth. Particularly, we use two of the holdout model runs to perform two synthetic calibrations. That is, we treat each of the two holdout model runs as the true data. Hence, we exclude the simulator discrepancy portion of the model. The goal is to see if the parameters used to generate the model runs can be reasonably identified. One- and two-way marginal posterior distributions for the six continuous parameters are shown in Figure 5. These show that we are able to learn five of the six parameters well in one case (Synthetic Calibration 2). However, we are unable to learn the altitude from these data. This is not surprising, given that our sensitivity analyses showed that altitude played little to no role in the emulator, so the output cannot constrain this particular input. Upon closer investigation, we found that the grid used in the simulations was too coarse to learn anything about the altitude parameter in the range that was specified *a priori* (see



(a) Synthetic Calibration 1

(b) Synthetic Calibration 2

Figure 5: Marginal posterior distributions of continuous parameters in two synthetic calibration problems where we calibrate to model run output. True values are marked with vertical lines and X's. 95% contours are shown in the two-way marginal plots.

Table 1). We were unable to learn as much in Synthetic Calibration 1, likely because, for that particular wind pattern and release location, much of the plume missed the sensors closest to the release. Hence, figuring out the release timing was difficult.

Regarding the five categorical parameters that are inputs to WRF, we are able to identify these well in the two synthetic examples (these results are not shown). Most posterior probability (86% and 87% in the two examples) is given to the particular combination of five variables that are the true settings. Other synthetic calibrations using holdout data have shown that the land surface model can be difficult to identify in some cases (results not shown).

5 Inferring the Source of a Diablo Canyon Release

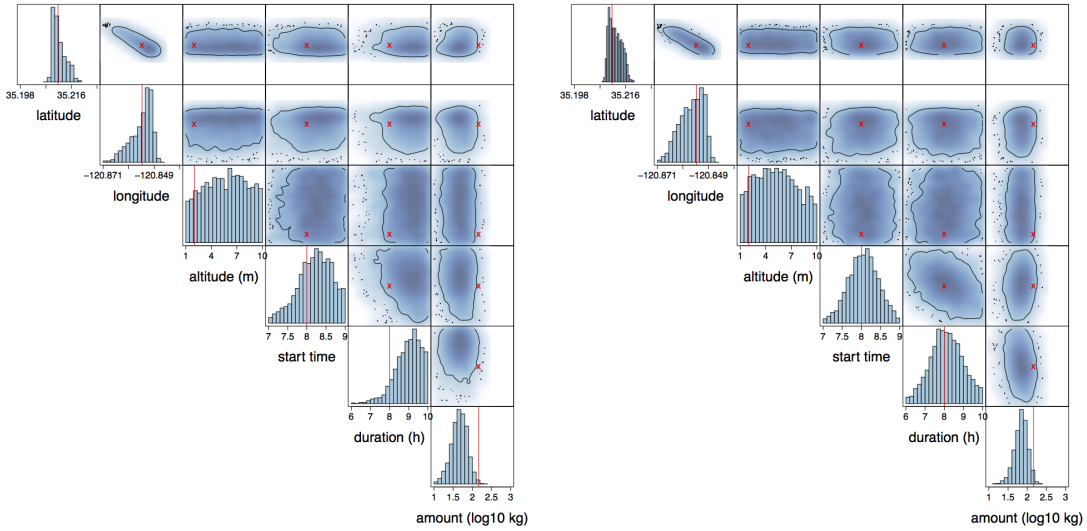
In this section, we present the results of the case study. We first discuss calibration results under two different discrepancy settings: (1) assuming no discrepancy (i.e., $\delta = 0$) and (2) using the modularized discrepancy discussed above. Figure 6 shows the one- and two-dimensional marginal posterior distributions of the continuous parameters for the no discrepancy and modularized

discrepancy cases. The true values of the parameters are also shown in Figure 6, from which we see that latitude and longitude are well identified under both discrepancy models. As can be seen from its nearly uniform posterior distribution, altitude is not well identified for the reason discussed in the previous section. The differences between these calibrated release location parameters are negligible under the two different discrepancy models. However, the release timing parameters (start time and duration) exhibit a fairly significant shift. The discrepancy model helps to reduce the bias of the timing parameters. While both models result in a biased amount parameter, the bias is reduced when a discrepancy model is included.

The bias in the calibrated amount parameter is most likely due to the fact that the amount is highly correlated with the maximum (over space and time) of a model run. The maximum is important because we are using the log scale, so other measurements may be orders of magnitude smaller and thus would not reflect much difference in the amount. Since the emulator is more smooth than the observations, the emulator maximum tends to be smaller than that seen in the observational data. Hence, we did not have a bias in our calibrated amount under the synthetic calibrations discussed earlier, as the synthetic data are more smooth, and maximum values tend to be well predicted by the emulator.

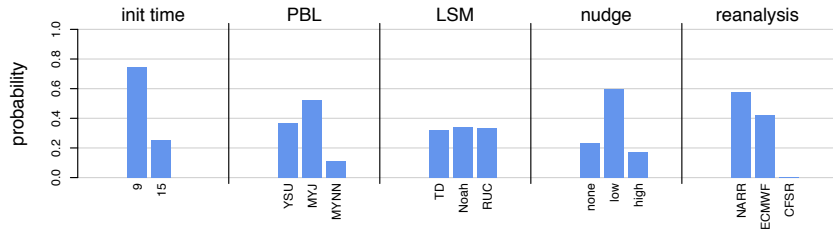
In Figure 7, the posterior marginal distributions of the categorical variables are shown for the two discrepancy cases. These distributions show significant change when a discrepancy model is used. The distribution of most parameters becomes less varied when the discrepancy is included. Notably, the reanalysis parameter strongly favors the North American model over the European model when the discrepancy is included. The distribution of the planetary boundary layer physics parameters is altered to give little mass to the YSU setting. There is also a much stronger preference for no nudging when discrepancy is included.

While the original goal of this analysis was to test the calibration methodology on an experimental release where many of the release characteristics were known, the field study was conducted over three decades ago at a time when global positioning satellites were not available. As a result, there is a discrepancy between the coordinates of the release location in the origi-

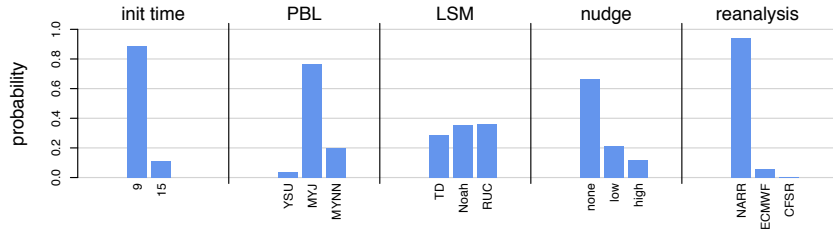


(a) Calibration Without Discrepancy (b) Calibration With Discrepancy

Figure 6: Marginal posterior distributions of continuous parameters obtained from calibrating to real data. True values are shown with vertical lines and X's. The left panel does not use discrepancy while the right panel does. 95% contours are shown in the two-way marginal plots.



(a) Calibration Without Discrepancy



(b) Calibration With Discrepancy

Figure 7: Marginal posterior distributions of categorical parameters obtained from calibrating to real data. The top panel does not use discrepancy while the bottom panel does. Unlike the continuous parameters, the true values of these parameters are unknown.

nal field study documentation (UTM 695368 Easting, 3898440 Northing, and zone 10) and the qualitative description of the release site in the paper by Thuillier (1992). Initially trusting the original field study documentation, we found that our estimate (posterior mean) of the location of the release (latitude and longitude) was biased by about 100 meters. Our estimate is more consistent with location described in Thuillier (1992), which indicates that the release occurred at the base of Containment Unit 2 (the southmost containment unit). We refer to the location described in Thuillier (1992) as the corrected release location because it is more probable under the calibration, as shown in Figure 8.

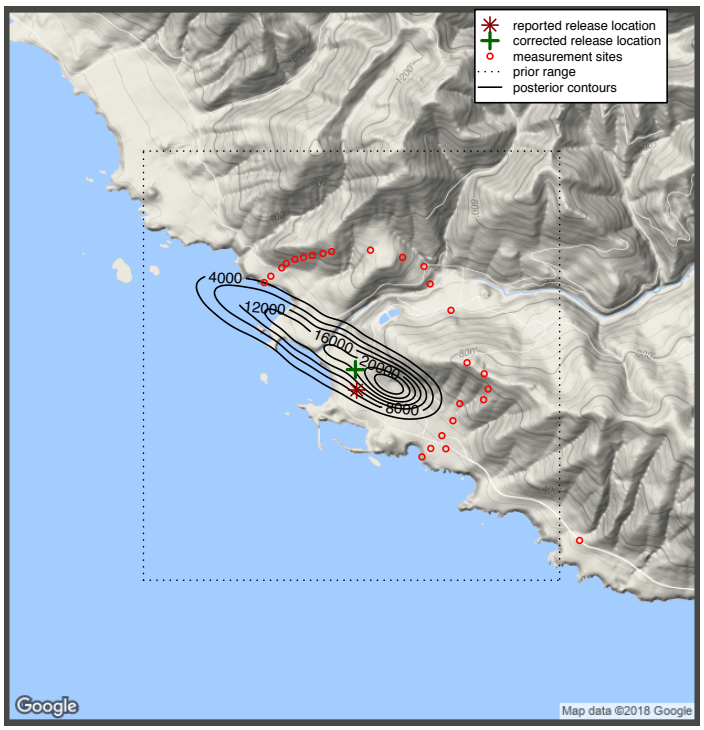


Figure 8: The marginal posterior distribution of the latitude and longitude parameters, as well as the reported location (see text for details).

As discussed previously, our calibration approach fixed the shape of the model discrepancy before inferring the calibration parameters. The scale parameter γ, which would inflate or deflate the effect, preferred a deflated discrepancy (95% probability interval of (0.58, 1.02) for γ). The model discrepancy inferred from the data can be a useful tool in understanding what parts of the

model could be explored further to improve predictive accuracy. We perform an exploratory data analysis of the discrepancy to try to determine its meaning. We see common shapes in groups of the discrepancy time series. When we cluster these time series, we partition our spatial field into areas where the discrepancy time series look similar. The clustered time series and the accompanying locations are shown in Figure 9. Clustering was performed using the fdakma R package (Parodi et al., 2015) with a K-means (but using the medians) type of algorithm that calculated the L2 distance between the first derivatives (approximated by differencing) of the time series. Thus, cluster membership was determined based on similarity of the shape, rather than amplitude of the curves. The number of clusters was selected based on visual assessment of the clustering results (i.e., do the clustered curves look similar in shape?). The more interesting parts of our discrepancy are in clusters two and five, shown in Figure 9. Cluster five corresponds to locations that would be in the early path of the plume under the meteorological conditions we observe. The shape of the time series in cluster five seems to indicate a discrepancy in the timing of the plume reaching those locations, implying that the simulator's early concentrations are too high and late concentrations are too low. This matches the fact that our inference regarding the timing parameters was improved when we included this discrepancy model. As shown in Figure 6, excluding a discrepancy results in later inferred start time. Cluster two corresponds to the spatial locations closest to the release. These have largest discrepancy likely because of the high concentration of the plume just after the release. With only a slight perturbation of the wind conditions, the early plume predictions can be in completely different locations because the plume is so concentrated. Thus, if wind predictions are only slightly inaccurate, the early plume predictions yield a large model discrepancy.

Our calibrated predictions with and without discrepancy for the locations in cluster five are shown in Figure 10. These show that including discrepancy brings a significant benefit at those locations while not being overly complex in shape. These also show that the extrapolated predictions including discrepancy can be fairly inaccurate. Indeed, the curves shown in Figure 9 show that the temporal extrapolation is linear. As in all discrepancy functions that are motivated

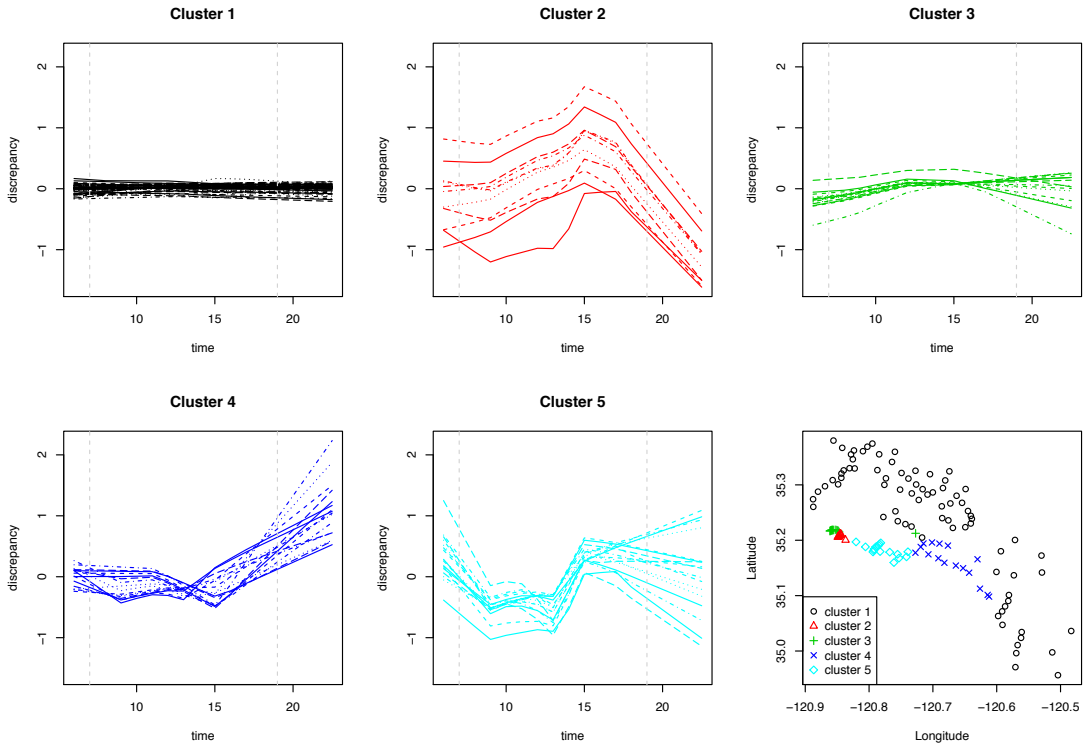


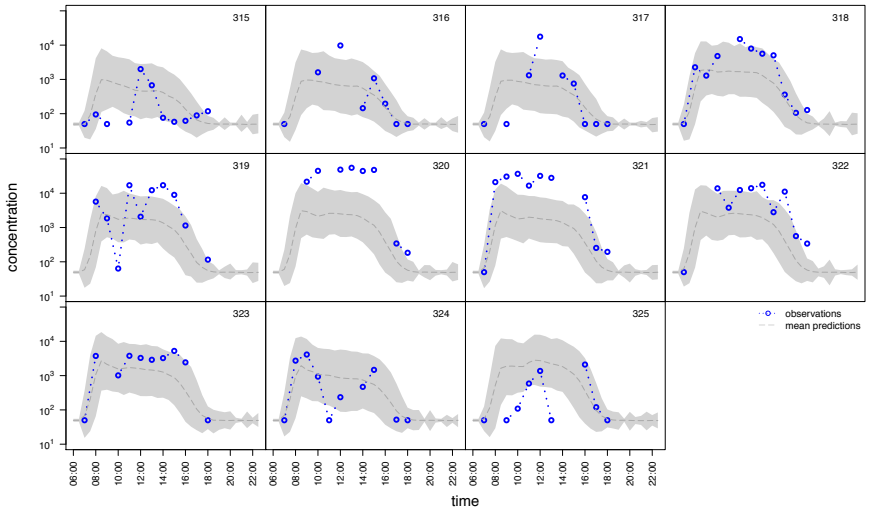
Figure 9: Clusters of discrepancy time series according to shape. The bottom right plot shows locations marked by cluster membership. The vertical dotted lines in the time series represent interval limits outside of which extrapolation begins.

by data rather than scientific input, this extrapolation should not be trusted.

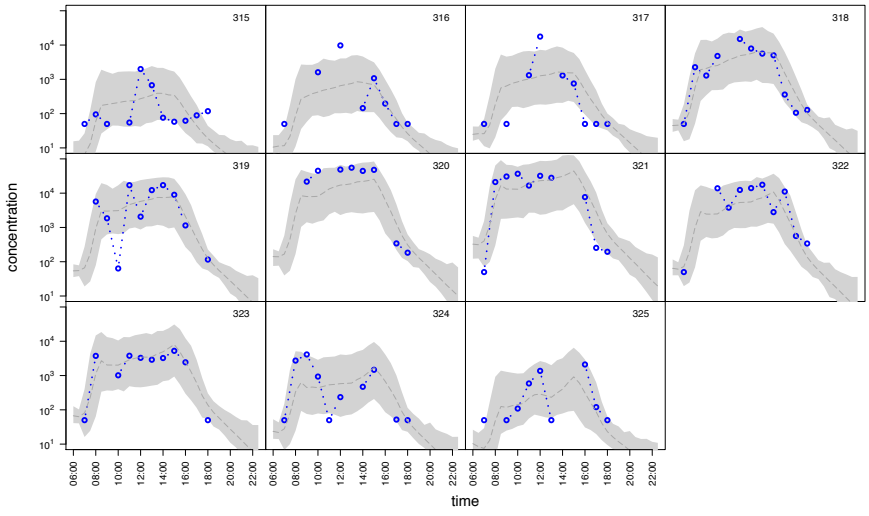
To see the broader effect of including discrepancy on calibrated prediction, we show the predicted (posterior mean) versus observed data in Figure 11. This shows that the discrepancy model, while simple, generally improves prediction. The improvement is most significant for large values, corresponding to the location clusters closest to the release.

6 Discussion

We have presented an analysis of a computer experiment with important applications to locating and assessing an atmospheric release. In the process, we have developed emulation methodology that can be scaled for use with large numbers of model runs when each has functional output. We have extended existing BMARS methodology to allow for categorical inputs and to model



(a) Prediction Excluding Discrepancy



(b) Prediction Including Discrepancy

Figure 10: Calibrated prediction for the locations in cluster 2. The top panel shows prediction excluding discrepancy, while the bottom panel includes discrepancy. The 95% pointwise probability intervals do not include the observational error, which is Normal with posterior standard deviation near 0.39 (95% probability interval (0.35, 0.43) for σ_F, symmetry coincidental).

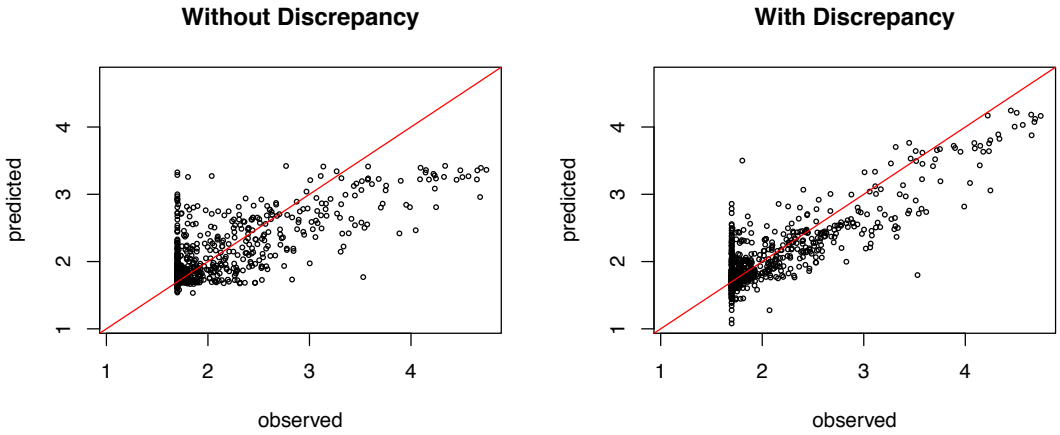


Figure 11: Calibrated prediction versus the observations, with and without the discrepancy in prediction.

in a reduced dimension space, and have shown how to use it for functional sensitivity analysis. In building this emulator, we have detailed how to keep track of different sources of uncertainty. We have extended modular approaches to computer model calibration for use with a BMARS emulator and a BMARS discrepancy function, paying special attention to identifiability.

The immediate applications of this work are for research and development purposes rather than emergency response. In an actual emergency, there would be limited time to run WRF (in forecast mode) to get the wind fields for use in FLEXPART. However, an approach that uses weather analogues (Delle Monache et al., 2011) rather than WRF runs may be feasible. An ensemble of FLEXPART simulations could be obtained in parallel, and emulation and calibration models could be fit thereafter. In order to be useful, emulation and calibration need to be done quickly and accurately. Hence, the emulation and calibration methodology outlined here has potential to be useful.

We acknowledge that there are many possible emulators to choose from, each with advantages and disadvantages. Because of the complexities of our input and output data in this case, the class of emulators is limited. BMARS in EOF space was a good emulator because of its reasonable predictive accuracy and its ability to be used for sensitivity analysis. The only other emulator that we could easily adapt for use in this case (handling continuous and categorical inputs, a

large number of simulations, and functional response while preserving a probabilistic sense of uncertainty) is Bayesian Additive Regression Trees (BART) (Chipman et al., 2010) used in EOF space. Our preliminary results show that BART requires training times that are comparable to those of BMARS, and can have very good predictive skills. As such, we believe that BART is a serious competitor to the models presented in this paper. In Section 3 of the supplementary material, we provide a brief comparison between BMARS as described in this paper, BART, and treed Gaussian processes (Gramacy and Lee, 2008) for a series of datasets.

A possible extension of the proposed model would be to include functional input. That is, rather than parameterizing WRF with five categorical parameters, we could include the entire spatio-temporal wind field as input for the emulator. This may be possible by decomposing wind fields onto a set of basis functions and including the basis function weights as inputs to the BMARS emulator.

7 Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 and was funded by Laboratory Directed Research and Development at LLNL under project tracking code PLS-14ERD006. The manuscript is released under UCRL number LLNL-JRNL-732282. The authors thank PG&E for access to the Diablo Canyon measurement data. The authors also thank Ronald Baskett and Philip Cameron-Smith from LLNL for helpful discussions about the simulations and measurement data.

8 Supplementary Material

Data (observations and simulations) and code are available in the online supplementary material.

References

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008), "Gaussian predictive process models for large spatial data sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70, 825–848.

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

Chakraborty, A., Mallick, B. K., Mcclarren, R. G., Kuranz, C. C., Bingham, D., Grosskopf, M. J., Rutter, E. M., Stripling, H. F., and Drake, R. P. (2013), "Spline-based emulators for radiative shock experiments with measurement error," *Journal of the American Statistical Association*, 108, 411–428.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010), "BART: Bayesian additive regression trees," *The Annals of Applied Statistics*, 4, 266–298.

Delle Monache, L., Nipen, T., Liu, Y., Roux, G., and Stull, R. (2011), "Kalman filter and analog schemes to postprocess numerical weather predictions," *Monthly Weather Review*, 139, 3554–3570.

Denison, D. G., Mallick, B. K., and Smith, A. F. (1998), "Bayesian MARS," *Statistics and Computing*, 8, 337–346.

Francom, D. (2017), *BASS: Bayesian Adaptive Spline Surfaces*, R package version 0.2.2.

Francom, D., and Sansó, B. (in press), "BASS: An R Package for Fitting and Performing Sensitivity Analysis of Bayesian Adaptive Spline Surfaces," *Journal of Statistical Software*.

Francom, D., Sansó, B., Kupresanin, A., and Johannesson, G. (2018), "Sensitivity Analysis and Emulation for Functional Data using Bayesian Adaptive Splines," *Statistica Sinica*, 28, 791–816.

Friedman, J. H. (1991a), "Estimating Functions of Mixed Ordinal and Categorical Variables Using Adaptive Splines," Tech. rep., DTIC Document.

—— (1991b), "Multivariate adaptive regression splines," *The annals of statistics*, 1–67.

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

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

Gramacy, R. B., Bingham, D., Holloway, J. P., Grosskopf, M. J., Kuranz, C. C., Rutter, E., Trantham, M., Drake, R. P., et al. (2015), "Calibrating a large computer experiment simulating radiative shock hydrodynamics," *The Annals of Applied Statistics*, 9, 1141–1168.

Gramacy, R. B., and Lee, H. K. H. (2008), "Bayesian treed Gaussian process models with an application to computer modeling," *Journal of the American Statistical Association*, 103, 1119–1130.

Green, P. J. (1995), "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, 82, 711–732.

Higdon, D. (1998), "A process-convolution approach to modelling temperatures in the North Atlantic Ocean," *Environmental and Ecological Statistics*, 5, 173–190.

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

Kaufman, C. G., Bingham, D., Habib, S., Heitmann, K., and Frieman, J. A. (2011), "Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology," *The Annals of Applied Statistics*, 2470–2492.

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

Liang, F., Paulo, R., Molina, G., Clyde, M. A., and Berger, J. O. (2008), "Mixtures of g priors for Bayesian variable selection," *Journal of the American Statistical Association*, 103.

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

Lucas, D. D., Simpson, M. D., Cameron-Smith, P., and Baskett, R. L. (2017), "Bayesian inverse modeling of the atmospheric transport and emissions of a controlled tracer release from a nuclear power plant," *Atmospheric Chemistry and Physics Discussions*, 2017, 1–36.

Ma, S., Racine, J. S., and Yang, L. (2015), "Spline regression in the presence of categorical predictors," *Journal of Applied Econometrics*, 30, 705–717.

Nott, D. J., Kuk, A. Y., and Duc, H. (2005), "Efficient sampling schemes for Bayesian MARS models with many predictors," *Statistics and Computing*, 15, 93–101.

Parodi, A., Patriarca, M., Sangalli, L., Secchi, P., Vantini, S., and Vitelli, V. (2015), *fdakma: Functional Data Analysis: K-Mean Alignment*, R package version 1.2.1.

Qian, P. Z. G., Wu, H., and Wu, C. J. (2008), "Gaussian process models for computer experiments with qualitative and quantitative factors," *Technometrics*, 50, 383–396.

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

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008), *Global sensitivity analysis: the primer*, John Wiley & Sons.

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Barker, D. M., Wang, W., and Powers, J. G. (2008), "A description of the advanced research WRF version 3," Tech. Rep. NCAR/TN-475+STR, National Center For Atmospheric Research Boulder Co Mesoscale and Microscale Meteorology Div.

Sobol', I. M. (1990), "On sensitivity estimation for nonlinear mathematical models," *Matematicheskoe Modelirovanie*, 2, 112–118.

—— (2001), "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates," *Mathematics and computers in simulation*, 55, 271–280.

Stohl, A., Forster, C., Frank, A., Seibert, P., and Wotawa, G. (2005), "Technical note: The Lagrangian particle dispersion model FLEXPART version 6.2," *Atmospheric Chemistry and Physics*, 5, 2461–2474.

Storlie, C. B., Lane, W. A., Ryan, E. M., Gattiker, J. R., and Higdon, D. M. (2015), "Calibration of computational models with categorical parameters and correlated outputs via Bayesian smoothing spline ANOVA," *Journal of the American Statistical Association*, 110, 68–82.

Stripling, H., McClarren, R., Kuranz, C., Grosskopf, M., Rutter, E., and Torralva, B. (2013), "A calibration and data assimilation method using the Bayesian MARS emulator," *Annals of Nuclear Energy*, 52, 103–112.

Thuillier, R. H. (1992), "Evaluation of a puff dispersion model in complex terrain," *Journal of the Air & Waste Management Association*, 42, 290–297.

United Nations Scientific Committee on the Effects of Atomic Radiation (2008), "Sources and effects of ionizing radiation. UNSCEAR 2008 report to the General Assembly, with scientific annex," Volume II.

Zellner, A., and Siow, A. (1980), "Posterior odds ratios for selected regression hypotheses," in *Bayesian Statistics: Proceedings of the First International Meeting Held in Valencia*, eds. J. M. Bernardo, M. H. DeGroot, D. V. L., and Smith, A. F. M., Valencia, Spain: University of Valencia Press, pp. 585–603.

Zhang, Y., Tao, S., Chen, W., and Apley, D. W. (2018), "A Latent Variable Approach to Gaussian Process Modeling with Qualitative and Quantitative Factors," *arXiv preprint arXiv:1806.07504*.

Inferring Atmospheric Release Characteristics in a Large Computer Experiment using Bayesian Adaptive Splines - Supplementary Material

Devin Francom[1], Bruno Sansó[2], Vera Bulaevskaya[3], Donald Lucas[4], Matthew Simpson[4]
[1]Los Alamos National Laboratory
[2]University of California Santa Cruz
[3]Climate Corporation
[4]Lawrence Livermore National Laboratory

December 7, 2018

1 BMARS Posterior Sampling

1.1 Likelihood

If we have $(y_i, \mathbf{x}_i)$ pairs for $i = 1, \ldots, N$, we model the relationship as

$$y_i = a_0 + \sum_{m=1}^{M} a_m B_m(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2) \tag{1}$$

$$B_m(\mathbf{x}) = \prod_{j=1}^{J_m} g_{jm}[s_{jm}(x_{v_{jm}} - t_{jm})]_+^{\alpha}, \quad g_{jm} = [(s_{jm}+1)/2 - s_{jm}t_{jm}]^{-\alpha} \tag{2}$$

where g is a scaling factor to make the basis function have maximum of one. Conditional on M and basis parameters, let $\mathbf{B}$ be the matrix of basis functions with a column of ones for the intercept.

1.2 Priors

For the number of basis functions, the weights, and the error variance we use

$$M|\lambda \sim Poisson(\lambda), \ \lambda \sim IG(h_1, h_2) \tag{3}$$

$$\mathbf{a}|\mathbf{B}, \sigma^2, \tau \sim N\left(\mathbf{0}, \frac{\sigma^2}{\tau}(\mathbf{B}'\mathbf{B})^{-1}\right), \ \sigma^2 \sim IG(g_1, g_2), \ \tau \sim Ga(b_1, b_2). \tag{4}$$

The default settings of $h_1 = h_2 = 10$ are fairly robust and suggested in Denison et al. (1998); Nott et al. (2005). The default settings of $g_1 = g_2 = 0$ with $b_1 = 1/2$ and $b_2 = N/2$ result in the Zellner-Siow Cauchy version of the g-prior (Liang et al., 2008). Sometimes it makes sense to give a lower bound to the prior for σ^2 in order to prevent overfitting. Then the lower bound is another parameter, but it is zero by default.

The knots, signs, variables, and degree of interaction (all of which are used to create $\mathbf{B}$) have a discrete uniform prior over the constrained space of possibilities, constrained to have each basis function have at least b non-zero values (to prevent to local fitting at the edges). The discrete uniform constant ends up being a necessary part of the RJMCMC acceptance ratio. Counting the number of possibilities with the constraint (to get the constant) is too difficult, so we use the unconstrained space constant as a conservative proxy. The unconstrained space has

$$\begin{aligned} s_{jm} &\in \{-1, 1\} && (5)\\ t_{jm}|v_{jm} &\in \{x_{1v_{jm}}, \ldots, x_{nv_{jm}}\} && (6)\\ v_{jm} &\in \{1, \ldots, p\} - \{v_{km}\}_{k \neq j} && (7)\\ J_m &\in \{1, \ldots, J_{\max}\} && (8) \end{aligned}$$

so that the constant is

$$c_m = \left(\frac{1}{2}\right)^{J_m} \left(\prod_{j=1}^{J_m} \frac{1}{n_{v_{jm}}}\right) \binom{p}{J_{\mathrm{m}}}^{-1} \left(\frac{1}{J_{\max}}\right). \tag{9}$$

This makes

$$\pi\left(\left\{J_m, \{s_{jm}, t_{jm}, v_{jm}\}_{j=1}^{J_m}\right\}_{m=1}^{M}\right) = \prod_{m=1}^{M} c_m 1(b_m > b). \tag{10}$$

1.3 Posterior

Then the posterior up to a constant is

$$\begin{aligned} \pi(M, \lambda, \mathbf{a}, \sigma^2, \tau, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}) \propto N\left(\mathbf{y}|\mathbf{B}\mathbf{a}, \sigma^2\mathbf{I}\right) N\left(\mathbf{a}|\mathbf{0}, \frac{\sigma^2}{\tau}(\mathbf{B}'\mathbf{B})^{-1}\right) Pois(M|\lambda) Ga(\lambda|h_1, h_2) M! \\ Ga(\tau|b_1, b_2) \prod_{m=1}^{M} \left(\frac{1}{2}\right)^{J_m} \left(\prod_{j=1}^{J_m} \frac{1}{n_{v_{jm}}}\right) \binom{p}{J_m}^{-1} \left(\frac{1}{J_{\max}}\right) 1(b_m > b). \end{aligned} \tag{11}$$

where the $M!$ accounts for all the ways you can get the M basis functions (ordering). Notice that we can rewrite $N\left(\mathbf{y}|\mathbf{Ba}, \sigma^2\mathbf{I}\right) N\left(\mathbf{a}|\mathbf{0}, \frac{\sigma^2}{\tau}(\mathbf{B}'\mathbf{B})^{-1}\right)$ as

$$(2\pi\sigma^2)^{-N/2}(2\pi\sigma^2/\tau)^{-(M+1)/2}|\mathbf{B}'\mathbf{B}|^{1/2}\exp\left\{\frac{-1}{2\sigma^2}\underbrace{[(\mathbf{y}-\mathbf{Ba})'(\mathbf{y}-\mathbf{Ba})+\tau\mathbf{a}'\mathbf{B}'\mathbf{Ba}]}_{\mathbf{y}'\mathbf{y}-2\mathbf{y}'\mathbf{Ba}+(1+\tau)\mathbf{a}'\mathbf{B}'\mathbf{Ba}}\right\} \tag{12}$$

$$= (2\pi\sigma^2)^{-N/2}(2\pi\sigma^2/\tau)^{-(M+1)/2}|\mathbf{B}'\mathbf{B}|^{1/2}\exp\left\{\frac{-(1+\tau)}{2\sigma^2}\left(\mathbf{a}-\frac{\hat{\mathbf{a}}}{1+\tau}\right)'\mathbf{B}'\mathbf{B}\left(\mathbf{a}-\frac{\hat{\mathbf{a}}}{1+\tau}\right)\right\}$$
$$\exp\left\{\frac{-1}{2\sigma^2}\left[\mathbf{y}'\mathbf{y}-\frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{B}\hat{\mathbf{a}}\right]\right\} \tag{13}$$

$$= (2\pi\sigma^2)^{-N/2}(2\pi\sigma^2/\tau)^{-(M+1)/2}|\mathbf{B}'\mathbf{B}|^{1/2}N\left(\mathbf{a}\middle|\frac{\hat{\mathbf{a}}}{1+\tau},\frac{\sigma^2}{1+\tau}(\mathbf{B}'\mathbf{B})^{-1}\right)\left(\frac{2\pi\sigma^2}{1+\tau}\right)^{(M+1)/2}|\mathbf{B}'\mathbf{B}|^{-1/2}$$
$$\exp\left\{\frac{-1}{2\sigma^2}\left[\mathbf{y}'\mathbf{y}-\frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{B}\hat{\mathbf{a}}\right]\right\} \tag{14}$$

$$= (2\pi\sigma^2)^{-N/2}\left(\frac{\tau}{1+\tau}\right)^{(M+1)/2}N\left(\mathbf{a}\middle|\frac{\hat{\mathbf{a}}}{1+\tau},\frac{\sigma^2}{1+\tau}(\mathbf{B}'\mathbf{B})^{-1}\right)\exp\left\{\frac{-1}{2\sigma^2}\left[\mathbf{y}'\mathbf{y}-\frac{1}{1+\tau}\underbrace{\hat{\mathbf{a}}'\mathbf{B}'\mathbf{B}\hat{\mathbf{a}}}_{\hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}}\right]\right\} \tag{15}$$

where $\hat{\mathbf{a}} = (\mathbf{B}'\mathbf{B})^{-1}\mathbf{B}'\mathbf{y}$. This shows that we can marginalize over $\mathbf{a}$. We could also marginalize over σ^2.

1.4 Reversible Jump MCMC

Our RJMCMC algorithm allows three types of moves, chosen with equal probability: Birth (add a basis function), death (delete a basis function), and change (move a knot/sign). We use these moves to sample $[M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda]$. In these moves $\mathbf{a}$ is marginalized out, which means that we do not need to worry about a transdimensional proposal for $\mathbf{a}$. Then we can sample $[\sigma^2|\mathbf{y}, M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}, \tau, \lambda]$. If we then sample $[\mathbf{a}|\cdot]$, we can sample $[\tau|\cdot]$.

1.4.1 Birth

Let $b(M^C \to M^P)$ be the proposal probability of going from the current model to the proposed model (which adds one basis function). We sample a proposal by drawing an interaction order $J_{M+1} \sim I(M^C)$, a set of J_{M+1} variables (without replacement) from $z(M^C)$, and corresponding knots and signs from their respective (unconstrained) priors. Here, I is a discrete probability mass function that puts weight on interaction orders according to how often they are used in the current model (with a constant w_1 added, so none are zero). The pmf z is similar but for the variables used in the current model and with constant w_2. Thus

$$b(M^C \to M^P) = P_{\text{birth}} I(J_{M+1}|M^C) z(\mathbf{v}_{M+1}|M^C)\left(\frac{1}{2}\right)^{J_{M+1}}\prod_{j=1}^{J_{M+1}}\frac{1}{n_{v_{jM+1}}} \tag{16}$$

and the acceptance probability is the maximum of one and

$$\alpha_{\text{birth}} = \frac{[M+1, \mathbf{J}^*, \mathbf{s}^*, \mathbf{t}^*, \mathbf{v}^*|\mathbf{y}, \sigma^2, \tau, \lambda] b(M^P \to M^C)}{[M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda] b(M^C \to M^P)} \tag{17}$$

and $b(M^P \to M^C) = P_{\text{death}} \frac{1}{M+1}$, as a basis function is chosen at random to kill.

$$[M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda] \propto \left(\frac{\tau}{1+\tau}\right)^{(M+1)/2} \exp\left\{\frac{-1}{2\sigma^2}\left[\mathbf{y}'\mathbf{y} - \frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\} Pois(M|\lambda)M!$$
$$\prod_{m=1}^{M}\left(\frac{1}{2}\right)^{J_m}\left(\prod_{j=1}^{J_m}\frac{1}{n_{v_{jm}}}\right)\binom{p}{J_m}^{-1}\left(\frac{1}{J_{\max}}\right)1(b_m > b) \tag{18}$$
$$\propto \left(\frac{\tau}{1+\tau}\right)^{(M+1)/2} \exp\left\{\frac{1}{2\sigma^2}\left[\frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\}\lambda^M$$
$$\prod_{m=1}^{M}\left(\frac{1}{2}\right)^{J_m}\left(\prod_{j=1}^{J_m}\frac{1}{n_{v_{jm}}}\right)\binom{p}{J_m}^{-1}\left(\frac{1}{J_{\max}}\right)1(b_m > b) \tag{19}$$

Now if $\mathbf{B}^*$ and $\hat{\mathbf{a}}^*$ are the candidate set of basis functions and least-squares weights,

$$\frac{[M+1, \mathbf{J}^*, \mathbf{s}^*, \mathbf{t}^*, \mathbf{v}^*|\mathbf{y}, \sigma^2, \tau, \lambda]}{[M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda]} = \left(\frac{\tau}{1+\tau}\right)^{1/2} \exp\left\{\frac{1}{2\sigma^2(1+\tau)}\left[\hat{\mathbf{a}}^*\mathbf{B}^{*\prime}\mathbf{y} - \hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\}\lambda$$
$$\left(\frac{1}{2}\right)^{J_{M+1}}\left(\prod_{j=1}^{J_{M+1}}\frac{1}{n_{v_{jM+1}}}\right)\binom{p}{J_{M+1}}^{-1}\left(\frac{1}{J_{\max}}\right)1(b_{M+1} > b) \tag{20}$$

and

$$\alpha_{\text{birth}} = \left(\frac{\tau}{1+\tau}\right)^{1/2} \exp\left\{\frac{1}{2\sigma^2(1+\tau)}\left[\hat{\mathbf{a}}^*\mathbf{B}^{*\prime}\mathbf{y} - \hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\}\lambda$$
$$\binom{p}{J_{M+1}}^{-1}\left(\frac{1}{J_{\max}}\right)1(b_{M+1} > b)\frac{P_{\text{death}}/(M+1)}{P_{\text{birth}}I(J_{M+1}|M^C)z(\mathbf{v}_{M+1}|M^C)} \tag{21}$$

1.4.2 Death

The death step is largely the reciprocal of the birth step:

$$\alpha_{\text{death}} = \frac{[M-1, \mathbf{J}^*, \mathbf{s}^*, \mathbf{t}^*, \mathbf{v}^*|\mathbf{y}, \sigma^2, \tau, \lambda]b(M^C \to M^P)}{[M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda]b(M^P \to M^C)} \tag{22}$$
$$= \left(\frac{\tau}{1+\tau}\right)^{-1/2} \exp\left\{\frac{1}{2\sigma^2(1+\tau)}\left[\hat{\mathbf{a}}^*\mathbf{B}^{*\prime}\mathbf{y} - \hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\}(1/\lambda)$$
$$\binom{p}{J_m}J_{\max}\frac{MP_{\text{birth}}I(J_m|M^P)z(\mathbf{v}_m|M^P)}{P_{\text{death}}}. \tag{23}$$

1.4.3 Change

Since there is no dimension change, the proposals and priors cancel resulting in

$$\alpha_{\text{change}} = \frac{[M, \mathbf{J}, \mathbf{s}^*, \mathbf{t}^*, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda]}{[M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}|\mathbf{y}, \sigma^2, \tau, \lambda]} \tag{24}$$
$$= \exp\left\{\frac{1}{2\sigma^2(1+\tau)}\left[\hat{\mathbf{a}}^*\mathbf{B}^{*\prime}\mathbf{y} - \hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\}. \tag{25}$$

1.5 Gibbs Steps

$$[\sigma^2|\mathbf{y}, M, \mathbf{J}, \mathbf{s}, \mathbf{t}, \mathbf{v}, \tau, \lambda] \propto IG(\sigma^2|g_1, g_2)(2\pi\sigma^2)^{-N/2} \exp\left\{\frac{-1}{2\sigma^2}\left[\mathbf{y}'\mathbf{y} - \frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right\} \tag{26}$$

$$\propto (\sigma^2)^{-N/2-g_1-1} \exp\left\{\frac{-1}{\sigma^2}\left[g_2 + \frac{1}{2}\left(\mathbf{y}'\mathbf{y} - \frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right)\right]\right\} \tag{27}$$

$$\sim IG\left(N/2 + g_1, g_2 + \frac{1}{2}\left[\mathbf{y}'\mathbf{y} - \frac{1}{1+\tau}\hat{\mathbf{a}}'\mathbf{B}'\mathbf{y}\right]\right) \tag{28}$$

$$[\mathbf{a}|\cdot] \sim N\left(\frac{\hat{\mathbf{a}}}{1+\tau}, \frac{\sigma^2}{1+\tau}(\mathbf{B}'\mathbf{B})^{-1}\right) \tag{29}$$

$$[\tau|\cdot] \propto N\left(\mathbf{a}|\mathbf{0}, \frac{\sigma^2}{\tau}(\mathbf{B}'\mathbf{B})^{-1}\right) Ga(\tau|b_1, b_2) \tag{30}$$

$$\propto \tau^{(M+1)/2+b_1-1} \exp\left\{-\tau\left[b_2 + \frac{1}{2\sigma^2}\mathbf{a}'\mathbf{B}'\mathbf{B}\mathbf{a}\right]\right\} \tag{31}$$

$$\sim Ga\left((M+1)/2 + b_1, b_2 + \frac{1}{2\sigma^2}\mathbf{a}'\mathbf{B}'\mathbf{B}\mathbf{a}\right) \tag{32}$$

2 Sobol' Decomposition

If we have a function $f(\mathbf{x})$ where $\mathbf{x} = (x_1, \ldots, x_p)$, we decompose it into

$$f(\mathbf{x}) = f_0 + \sum_{i=1}^{p} f_i(x_i) + \sum_{i=1}^{p}\sum_{j>i} f_{ij}(x_i, x_j) + \cdots + f_{1\ldots p}(x_1, \ldots, x_p) \tag{33}$$

where all terms added above are orthogonal. In addition, all terms except f_0 are centered at zero. This is achieved by building each term such that

$$f_0 = \int f(\mathbf{x})d\mathbf{x} \tag{34}$$

$$f_i(x_i) = \int f(\mathbf{x})d\mathbf{x}_{-i} - f_0 \tag{35}$$

$$f_{ij}(x_i, x_j) = \int f(\mathbf{x})d\mathbf{x}_{-ij} - f_i(x_i) - f_j(x_j) - f_0 \tag{36}$$

as so on, where integrals are over the bounds of each x_i. Without loss of generality, we assume that the bounds are zero and one.

We are interested in partitioning $Var(f(\mathbf{x}))$ into variance from each main effect and interaction. First, note that if we assume that x_i is a random variable with uniform distribution $f_i(x_i) = E(f(\mathbf{x})|x_i) - f_0$ and $f_0 = E(f(\mathbf{x}))$. Also note that $Var(f(\mathbf{x})) = E(f^2(\mathbf{x})) - E(f(\mathbf{x}))^2$. Now, squaring Equation 33 and integrating, we obtain

$$E(f^2(\mathbf{x})) = f_0^2 + \sum_{i=1}^{p}\int f_i^2(x_i)dx_i + \sum_{i=1}^{p}\sum_{j>i}\int f_{ij}^2(x_i, x_j)dx_i dx_j + \cdots + \int f_{1\ldots p}^2(x_1, \ldots, x_p)d\mathbf{x}$$

which lacks any crossproduct terms because all the terms are orthogonal. We could also write

each term above as a variance, i.e., $Var(f_i(x_i)) = \int f_i^2(x_i)dx_i - 0$. We subtract 0 because $(\int f_i(x_i)dx_i)^2 = 0$. This is the case for each term. Thus,

$$E(f^2(\mathbf{x})) = f_0^2 + \sum_{i=1}^{p} Var(f(x_i)) + \sum_{i=1}^{p}\sum_{j>i} Var(f(x_i, x_j)) + \cdots + Var(f_{1\ldots p}(x_1, \ldots, x_p))$$

and we have that

$$Var(f(\mathbf{x})) = \sum_{i=1}^{p} Var(f(x_i)) + \sum_{i=1}^{p}\sum_{j>i} Var(f(x_i, x_j)) + \cdots + Var(f_{1\ldots p}(x_1, \ldots, x_p)),$$

thus decomposing the variance of f into variance due to each main effect and interaction. Note that the way we construct the main effects and interactions in Equations 34 through 36 is sequential, so that a two way interaction functions is the effect after taking into account the two main effects.

We outline some practical considerations for obtaining the variance decomposition above, as discussed in Chen et al. (2005). First, if $\mathbf{u} \subseteq \{1, \ldots, p\}$ of size s and $\mathbf{x_u} = \{x_{u_1}, \ldots, x_{u_s}\}$, then we write the effect $f_u(\mathbf{x_u})$ (interaction if $s > 1$, main effect if $s = 1$) can be written as

$$f_u(\mathbf{x_u}) = \hat{f}_u(\mathbf{x_u}) - \sum_{\mathbf{v}\in\{P(\mathbf{u})-\mathbf{u}-\emptyset\}} f_v(\mathbf{x_v}) - f_0$$
$$\hat{f}_u(\mathbf{x_u}) = \int f(\mathbf{x})d\mathbf{x}_{-\mathbf{u}}$$

where $P(\mathbf{u})$ is the power set of $\mathbf{u}$. Note that $\hat{f}_u$ denotes the non-centered version of f_u. This recursive definition gives rise (with some algebra) to a simpler formulation only in terms of the non-centered effects,

$$f_u(\mathbf{x_u}) = \sum_{\mathbf{v}\in P(\mathbf{u})} (-1)^{|\mathbf{u}|-|\mathbf{v}|} \hat{f}_v(\mathbf{x_v})$$

where we define $\hat{f}_v(\mathbf{x_v}) = f_0$ if $\mathbf{v} = \emptyset$. With some further algebra, we can see that

$$Var(f_u(\mathbf{x_u})) = \sum_{\mathbf{v}\in\{P(\mathbf{u})-\emptyset\}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} Var\left(\hat{f}_v(\mathbf{x_v})\right)$$

and it is easy to show that

$$Var\left(\hat{f}_u(\mathbf{x_u})\right) = \int \hat{f}_u^2(\mathbf{x_u})d\mathbf{x_u} - f_0^2. \tag{37}$$

Thus, if we can evaluate Equation 37 analytically, we can obtain the variance decomposition analytically.

2.1 BASS Sobol' Decomposition

The Sobol' decomposition for the BASS model can be done analytically. The details for obtaining the Sobol' decomposition and functional Sobol' decomposition when inputs are continuous are in Francom et al. (2018). Here, we will describe the details when we have categorical inputs. We will also describe how to get functional sensitivity indices when we use the EOF approach to emulation described in this paper.

First, we review (Chen et al., 2005) that when we use a tensor product basis function approach such that $f(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{j=1}^{J_m} h_{jm}(x_{v_{jm}})$, the quantities we need to obtain have to do with the integral of h. To simplify the notation, and without loss of generality, rewrite the previously stated function in terms of interactions between all the variables rather than only the variables active in the particular basis function $f(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{v=1}^{p} h_{vm}(x_v)$. Then the integrals required are $C^1_{vm} = \int h_{vm}(x_v)dx$ and $C^2_{vm_1m_2} = \int h_{vm_1}(x_v)h_{vm_2}(x_v)dx_v$. These quantities are discussed when inputs are continuous in Francom et al. (2018). For categorical inputs we use sums rather than integrals, so that

$$C^1_{vm} = \frac{1}{|D_v|} \sum_{z \in D_v} 1(z \in D_{vm}) = \frac{|D_{vm}|}{|D_v|}$$
$$C^2_{vm_1m_2} = \frac{1}{|D_v|} \sum_{z \in D_v} 1(z \in D_{vm_1})1(z \in D_{vm_2}) = \frac{|D_{vm_1} \cap D_{vm_2}|}{|D_v|}$$

where D_v is the set of all categories of variable v and D_{vm} is the subset of categories of v used in basis function m. Now if $\mathbf{u}$ is a set of variable indices,

$$Var(\hat{f}_u(\mathbf{x_u})) = \sum_{m_1=1}^{M} \sum_{m_2=1}^{M} a_{m_1} a_{m_2} \left(\prod_{l \notin \mathbf{u}} C^1_{lm_1} C^1_{lm_2} \right) \left(\prod_{l \in \mathbf{u}} C^2_{lm_1m_2} \right) - f_0^2.$$

2.2 Functional Sobol' Decomposition - EOFs

While the functional Sobol' decomposition of a functional BASS model is described in Francom et al. (2018), the model we use is slightly different. We use BASS for modeling weights for EOFs. To get a functional Sobol' decomposition in this case, we need to do a little more work.

First, consider the case where we are interested in getting Sobol' indices for the functional variables like we do the other variables. The function we are decomposing is $f(\mathbf{r}, \mathbf{x}) = \sum_{i=1}^{k} K_i(\mathbf{r}) w_i(\mathbf{x})$ where $\mathbf{r} = (s, t)$. If we define

$$\hat{w}_i^u(\mathbf{x_u}) = \int w_i(\mathbf{x}) d\mathbf{x_{-u}}$$
$$K_i = \int K_i(\mathbf{r}) d\mathbf{r}$$

then we can obtain the overall mean

$$f_0 = \int \int \sum_{i=1}^{k} K_i(\mathbf{r}) w_i(\mathbf{x}) d\mathbf{r} d\mathbf{x} = \sum_{i=1}^{k} \int K_i(\mathbf{r}) d\mathbf{r} \int w_i(\mathbf{x}) d\mathbf{x}$$
$$= \sum_{i=1}^{k} K_i w_i^0$$

and the non-centered effects as

$$\hat{f}_r(\mathbf{r}) = \sum_{i=1}^{k} K_i(\mathbf{r}) w_i^0$$
$$\hat{f}_u(\mathbf{x_u}) = \sum_{i=1}^{k} K_i \hat{w}_i^u(\mathbf{x_u})$$
$$\hat{f}_{ru}(\mathbf{r}, \mathbf{x_u}) = \sum_{i=1}^{k} K_i(\mathbf{r}) \hat{w}_i^u(\mathbf{x_u}).$$

Then we need to obtain

$$\begin{aligned} Var\left(\hat{f}_r(\mathbf{r})\right) &= \int \hat{f}_r^2(\mathbf{r}) d\mathbf{r} - f_0^2 \\ &= \sum_{i=1}^{k}\sum_{j=1}^{k} w_i^0 w_j^0 \int K_i(\mathbf{r}) K_j(\mathbf{r}) d\mathbf{r} - f_0^2 \end{aligned} \tag{38}$$

$$\begin{aligned} Var\left(\hat{f}_u(\mathbf{x_u})\right) &= \int \hat{f}_u^2(\mathbf{x_u}) d\mathbf{x_u} - f_0^2 \\ &= \sum_{i=1}^{k}\sum_{j=1}^{k} K_i K_j \int \hat{w}_i^u(\mathbf{x_u}) \hat{w}_j^u(\mathbf{x_u}) d\mathbf{x_u} - f_0^2 \end{aligned} \tag{39}$$

$$\begin{aligned} Var\left(\hat{f}_{ru}(\mathbf{r}, \mathbf{x_u})\right) &= \int\int \hat{f}_{ru}^2(\mathbf{r}, \mathbf{x_u}) d\mathbf{r} d\mathbf{x_u} - f_0^2 \\ &= \sum_{i=1}^{k}\sum_{j=1}^{k} \int K_i(\mathbf{r}) K_j(\mathbf{r}) d\mathbf{r} \int \hat{w}_i^u(\mathbf{x_u}) \hat{w}_j^u(\mathbf{x_u}) d\mathbf{x_u} - f_0^2 \end{aligned} \tag{40}$$

which simplifies in Equation 38 and Equation 40 when the basis is orthogonal, but not in Equation 39. Then, the only quantity left to seek an analytical solution for is

$$\begin{aligned} \int \hat{w}_i^u(\mathbf{x_u}) \hat{w}_j^u(\mathbf{x_u}) d\mathbf{x_u} &= a_{0i} a_{0j} + a_{0i}(w_j^0 - a_{0j}) + a_{0j}(w_i^0 - a_{0i}) \\ &+ \sum_{m_i=1}^{M_i}\sum_{m_j=1}^{M_j} a_{mi} a_{mj} \left(\prod_{l \notin \mathbf{u}} C_{lmi}^1 C_{lmj}^1\right)\left(\prod_{l \in \mathbf{u}} C_{lmij}^2\right). \end{aligned} \tag{41}$$

Second, if we get Sobol' indices as functions of r, we have

$$f_0(r) = \sum_{i=1}^{k} K_i(r) w_i^0 \tag{42}$$
$$\hat{f}_u(\mathbf{x_u}) = \sum_{i=1}^{k} K_i(r) \hat{w}_i^u(\mathbf{x_u}) \tag{43}$$

and we need to obtain

$$Var(\hat{f}_u(\mathbf{x_u})) = \int \hat{f}_u^2(\mathbf{x_u}) d\mathbf{x_u} - f_0^2(r) \tag{44}$$

$$= \sum_{i=1}^{k} \sum_{j=1}^{k} K_i(r) K_j(r) \int \hat{w}_i^u(\mathbf{x_u}) \hat{w}_j^u(\mathbf{x_u}) d\mathbf{x_u} - f_0^2(r) \tag{45}$$

which again utilizes Equation 41.

3 Emulator Comparison

At the request of a reviewer, we include some discussion of how to choose an emulator. Choosing an emulator is more than just choosing the nonlinear (or possibly linear) regression model that predicts the best, though good prediction is important. If the emulator cannot be evaluated quickly, it is not useful. If it does not quantify its uncertainty, it can give unreliable results when used for other uncertainty quantification tasks, like calibration or sensitivity analysis. There are many qualitative characteristics that are important to building a good emulator. Below, we discuss important qualitative and quantitative characteristics to consider, and we evaluate a few possible emulators based on these characteristics for a few datasets. This is not meant to be a comprehensive comparison, but merely shows a limited comparison that indicates that BMARS is a suitable choice for our emulation problem.

We give particular attention to three possible emulators in the following sections: (1) treed Gaussian processes (TGP) (Gramacy and Lee, 2008), (2) BMARS as we have described in this paper, and (3) Bayesian additive regression trees (BART) (Chipman et al., 2010). We choose these three to compare because they can be used for data of the type that we have in our case study (i.e., continuous and categorical inputs, not necessarily data size). Other methods could be compared, as in Swiler et al. (2014), but a full comparison is beyond the scope of this work.

3.1 Qualitative Characteristics

Certain qualitative characteristics make some emulators more useful than others in particular situations. When quantitative accuracy is similar for different emulators, qualitative characteristics can help choose between them. In some cases, qualitative characteristics make some emulators impossible to use for given datasets. Below, we consider a number of qualitative characteristics to consider when choosing an emulator.

3.1.1 Data size

Emulators like the Gaussian process do not scale well. Traditionally this was less of a problem, as large numbers of computer model runs were less common. Today, large numbers of computer model runs are more feasible because of larger computers. In addition, as computers get larger, more complex models can be simulated. Naturally, as a model becomes a more complex function of parameters, an emulator will require more training data (more model runs) to be accurate. This puts users who would emulate with GPs, which have desirable properties but are not scalable, in the difficult position of wanting fewer model runs than what may be necessary to capture the model dynamics and, hence, to build an accurate emulator. The common question of why we need an emulator when we can get a large number of model runs misses the fact that the model runs are usually performed in parallel. Thus, while each individual model run may be

expensive in time, they can run simultaneously. All of the calibration approaches that we can imagine would require at least some sequential model runs.

BMARS and BART both scale linearly with the number of model runs. Scalable approximations to GP models are popular, as well. TGP becomes more scalable as the tree part of the model becomes more complex.

3.1.2 Stationarity

Typical GP models are stationary. This is powerful when the computer model is roughly stationary. In those cases, the GP can fit much better than non-stationary approaches, especially with small sample sizes.

When stationarity is a bad assumption, GP models will have difficulty. BMARS and BART have nonstationary mean functions. There are nonstationary GP approaches, but they are likely to be costly in other ways. TGPs can partition the input space into regions of stationarity, a powerful ability, though it comes at the expense of continuity. Another approach would be to use a GP with a nonstationary mean function, for instance, a GP with a BMARS mean function as in Chakraborty et al. (2013).

An element of the stationarity assumption has to do with global versus local fit. The traditional GP provides a global fit (global correlation lengths), where the BART model is a local fit (trees correspond to a partition of the input space). The BMARS model is something of a mixture, as some basis functions can be linear functions over most of the input space while other basis functions pertain to smaller portions of the input space.

3.1.3 Input/output types

Typical computer modeling scenarios have some continuous inputs and a continuous output. The case of multivariate output is well studied and presented in this paper (other references). Any nonlinear regression model could be applied in EOF space. However, when the inputs include categorical variables, there is additional complexity. For instance, incorporating categorical variables into a GP is not as natural incorporating them into tree models. TGP provides an approach for this, where different GP models using the continuous parameters can be used at the leaves of a tree model, which includes the categorical parameters.

BART and BMARS have more natural ways of incorporating categorical variables.

3.1.4 Degree of interaction

While the GP can model any degree of interaction theoretically, in practice, it will require large amounts of data to accurately model high order interactions. This is the best that can be expected from any emulator.

BMARS and BART both adaptively discover the degree of interaction (and similarly perform variable selection), though that degree is typically capped by the user.

3.1.5 Interpolation

The GP can be an interpolator, though most users opt to include a nugget for numerical stability and sometimes better predictive accuracy (Gramacy and Lee, 2012). The property of the GP that allows for it to interpolate, namely that the correlation of outputs depends of the distance between inputs, also results in its desirable heteroscedasticity. GP predictive uncertainty is larger when predicting farther from training inputs.

BMARS and BART are not interpolators and do not have the desirable heteroscedasticity property. However, the approach of Chakraborty et al. (2013), which is a GP with a BMARS

mean function, is an interpolator and has the heteroscedasticity property. The same could be done with a BART mean function.

For our case study, interpolation is less realistic, since we are modeling in EOF space. To interpolate, we would not be able to do dimension reduction.

3.1.6 Continuity

Continuity can be a strong and useful assumption. Many computer models are known to be continuous functions of the inputs. The GP has continuity and also possibly differentiability built into it. The treed GP is discontinuous, but when used to only tree on the categorical variables, continuity can be preserved.

BMARS is continuous, though typically not differentiable. BART, as with other tree-based models, is inherently discontinuous. This can result in poor fits for small sample sizes.

3.1.7 Uses

For the purposes of calibration, accurate prediction with appropriate prediction uncertainty is important. This makes most Bayesian nonlinear regression models possibly suitable for calibration. For the purposes of adaptive design, interpolation and the heteroscedasticity of the GP can be crucial. For the purposes of sensitivity analysis, most approaches would require Monte Carlo approximations of ANOVA decompositions. The BMARS approach has analytical ANOVA decomposition (Francom et al., 2018), and the GP has at least a partial analytical ANOVA decomposition in some cases (Oakley and O'Hagan, 2004).

3.2 Quantitative Characteristics

The question of which emulator will be most accurate certainly depends on the dataset. Here, we compare three emulation methods (TGP, BMARS, and BART) based on prediction performance (including uncertainty) for three datasets, each with both categorical and continuous inputs. The datasets are (1) a synthetic example from Gramacy et al. (2010); (2) the EOF weights described in this case study, for the first EOF; and (3) the weights for the 50th EOF. We examine both the 1st and 50th EOF weights because the earlier EOFs tend to have higher signal to noise ratio than the later ones.

3.2.1 Synthetic Example

The first dataset is generated from a function taken from Gramacy et al. (2010), Section 2. The function has 11 inputs and is of the form

$$f(\mathbf{x}) = \begin{cases} 10\sin(\pi x_1 x_2) & x_{11} = 1 \\ 20(x_3 - 0.5)^2 & x_{11} = 2 \\ 10x_4 + 5x_5 & x_{11} = 3 \\ 10x_1 + 5x_2 + 20(x_3 - 0.5)^2 + 10\sin(\pi x_4 x_5) & x_{11} = 4 \end{cases} \tag{46}$$

so that 10 of the inputs are continuous (though five are noise) and one is categorical. Standard Normal noise is added to the output. We generate three ensembles of "model runs" from this model with different samples sizes (n). We term the ensembles small ($n = 50$), medium ($n = 500$), and large ($n = 5000$). For each ensemble, we fit the emulator and predict the output at 1000 new inputs. We also obtain a 90% prediction interval for each of the 1000 points. We use these to assess empirical coverage and interval length. Finally, we keep track of the time that model

fitting requires. We repeat all of this around 40 times, each with a new ensemble of inputs. For reference, we also include the random forest prediction performance (with no attempt at quantifying uncertainty). We use default settings for all but the TGP approach, for which we mimic the best values used in Gramacy et al. (2010) (page 8). We exclude the TGP fit for the large ensemble case, where it takes prohibitively long to run.

Figure 1 shows the results of this simulation. We see that, for the small datasets ($n = 50$), there is not a lot of variation in the RMSE, but that BMARS can have bad coverage. This is likely a result of BMARS overfitting, which could be addressed by changing the parameters. We expected TGP to perform best for the small data case, but it could be that there was not enough data to sufficiently grow the tree model. This is a case when the small number of model runs is too few to capture some important dynamics in the model. With more data ($n = 500$), TGP performs much better, and the three Bayesian methods have fairly good coverage. In the large data case ($n = 5000$), TGP would undoubtedly perform very well, but takes prohibitively long. In that case, BMARS performs best, and both BMARS and BART perform better than the random forest for only moderately longer training time. It is likely that there exist other implementations of the random forest that are faster.

3.2.2 Diablo Canyon Simulations - EOF 1

In this example, we opt for data that is more informative to this case study. We use the first EOF weight, where EOFs are calculated using the full dataset. The EOF weights are then subsampled to sample sizes of 50, 500, and 5000 as in the previous example. (In a perfect world, we would have taken new LHS samples rather than subsampling a LHS design, but this was not the priority of the case study.) We train the three emulators using the subsampled data, repeating that process for about 40 randomly chosen subsamples of the specified sizes. Prediction accuracy is then assessed based on prediction performance of 1000 held out samples. Results are shown in Figure 2.

For small datasets ($n = 50$) we see little variation in accuracy except for the poor coverage of BMARS, again likely because of overfitting. Here, TGP does not see as substantial an improvement in prediction RMSE as the data size increases, even as we adjusted the tree parameter priors. BMARS and BART see substantial improvement as the data size increases, and have similar predictive accuracy.

We note that deciding the number of model runs necessary for building a suitable emulator is nontrivial. Various rules of thumb exist based on strong assumptions, but often the scalability of the GP plays a role. GP users often prefer not having large numbers of model runs because of the scalability issues associated with the GP. This comes contrary to the notion in statistics that more (good) data is always better. The fact that in all the examples we present here, the BMARS model with large data outperforms the TGP model with medium data reinforces the idea that a non-GP emulator trained with more data is better than a GP emulator trained with less data. Similarly, there is some point where BMARS and BART will not be able to handle a dataset and something like nearest neighbors will perform better with more data than more complex models with less data. It is our opinion that, barring design considerations, the model runs should be used to choose the emulator rather than that the emulator should be used to choose the model runs. There are further practical considerations regarding how accurate an emulator needs to be, but those are more case-specific.

3.2.3 Diablo Canyon Simulations - EOF 50

This example is similar to the previous one but uses the 50th EOF weight rather than the first EOF weight. We expect that the later weights in the decomposition will be noisier than the

early weights, as they are more apt to pick up seemingly random variations. Results are shown in Figure 3.

In this example, we see more constant performance from all the methods, again with improvement as more data are used for training.

References

Chakraborty, A., Mallick, B. K., Mcclarren, R. G., Kuranz, C. C., Bingham, D., Grosskopf, M. J., Rutter, E. M., Stripling, H. F., and Drake, R. P. (2013), "Spline-based emulators for radiative shock experiments with measurement error," *Journal of the American Statistical Association*, 108, 411–428.

Chen, W., Jin, R., and Sudjianto, A. (2005), "Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty," *Journal of mechanical design*, 127, 875–886.

Chipman, H. A., George, E. I., McCulloch, R. E., et al. (2010), "BART: Bayesian additive regression trees," *The Annals of Applied Statistics*, 4, 266–298.

Denison, D. G., Mallick, B. K., and Smith, A. F. (1998), "Bayesian MARS," *Statistics and Computing*, 8, 337–346.

Francom, D., Sansó, B., Kupresanin, A., and Johannesson, G. (2018), "Sensitivity Analysis and Emulation for Functional Data using Bayesian Adaptive Splines," *Statistica Sinica*, 28, 791–816.

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

Gramacy, R. B., and Lee, H. K. H. (2008), "Bayesian treed Gaussian process models with an application to computer modeling," *Journal of the American Statistical Association*, 103, 1119–1130.

Gramacy, R. B., Taddy, M., et al. (2010), "Categorical inputs, sensitivity analysis, optimization and importance tempering with tgp version 2, an R package for treed Gaussian process models," *Journal of Statistical Software*, 33, 1–48.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., and Berger, J. O. (2008), "Mixtures of g priors for Bayesian variable selection," *Journal of the American Statistical Association*, 103.

Nott, D. J., Kuk, A. Y., and Duc, H. (2005), "Efficient sampling schemes for Bayesian MARS models with many predictors," *Statistics and Computing*, 15, 93–101.

Oakley, J. E., and O'Hagan, A. (2004), "Probabilistic sensitivity analysis of complex models: a Bayesian approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66, 751–769.

Swiler, L. P., Hough, P. D., Qian, P., Xu, X., Storlie, C., and Lee, H. (2014), "Surrogate models for mixed discrete-continuous variables," in *Constraint Programming and Decision Making*, Springer, pp. 181–202.

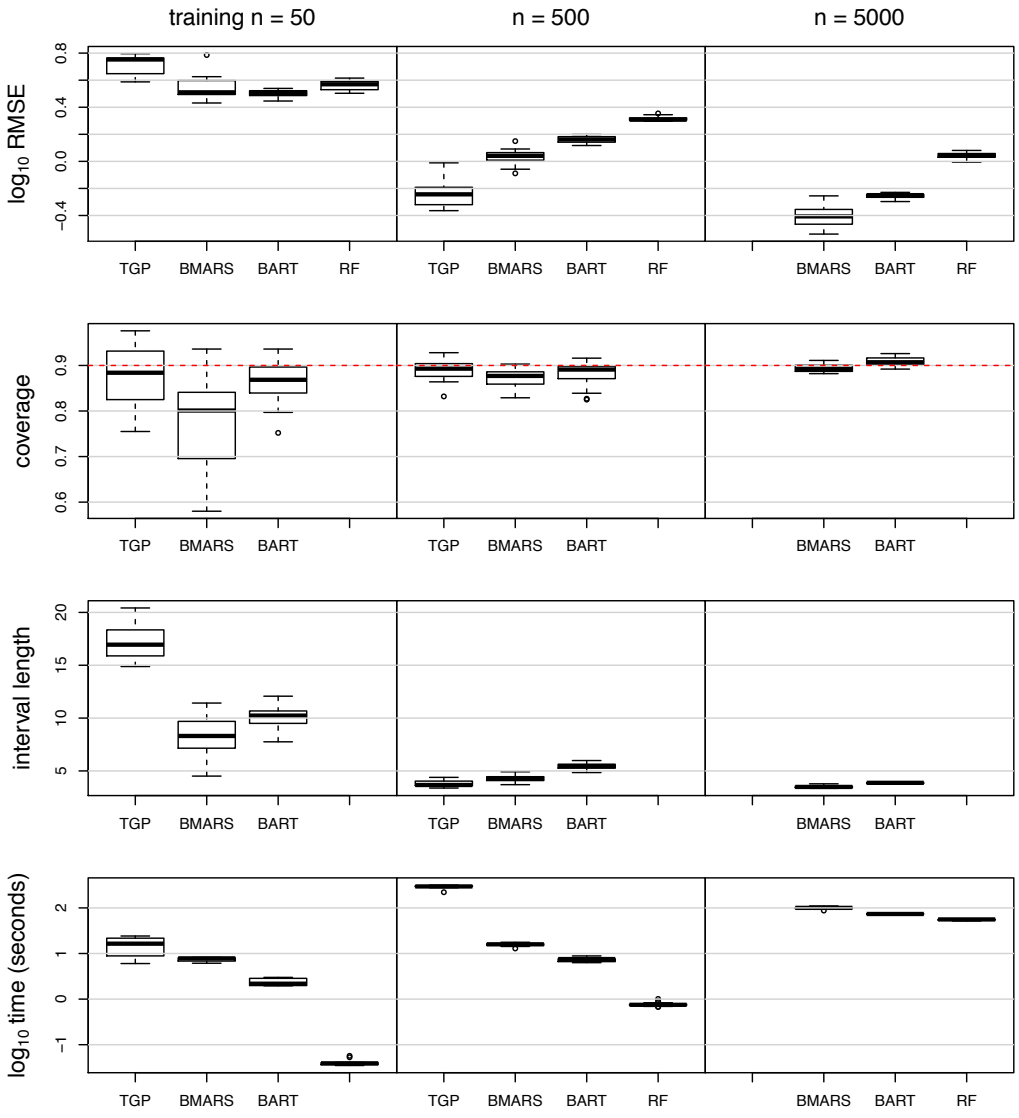


Figure 1: Comparison of emulators for the function given in Equation 46. The top row shows the root mean square error (RMSE) on the $\log_{10}$ scale. The second row shows the empirical coverage of 90% prediction intervals. The third row shows average 90% prediction interval length. The final row shows time that emulator training takes on the $\log_{10}$ scale.

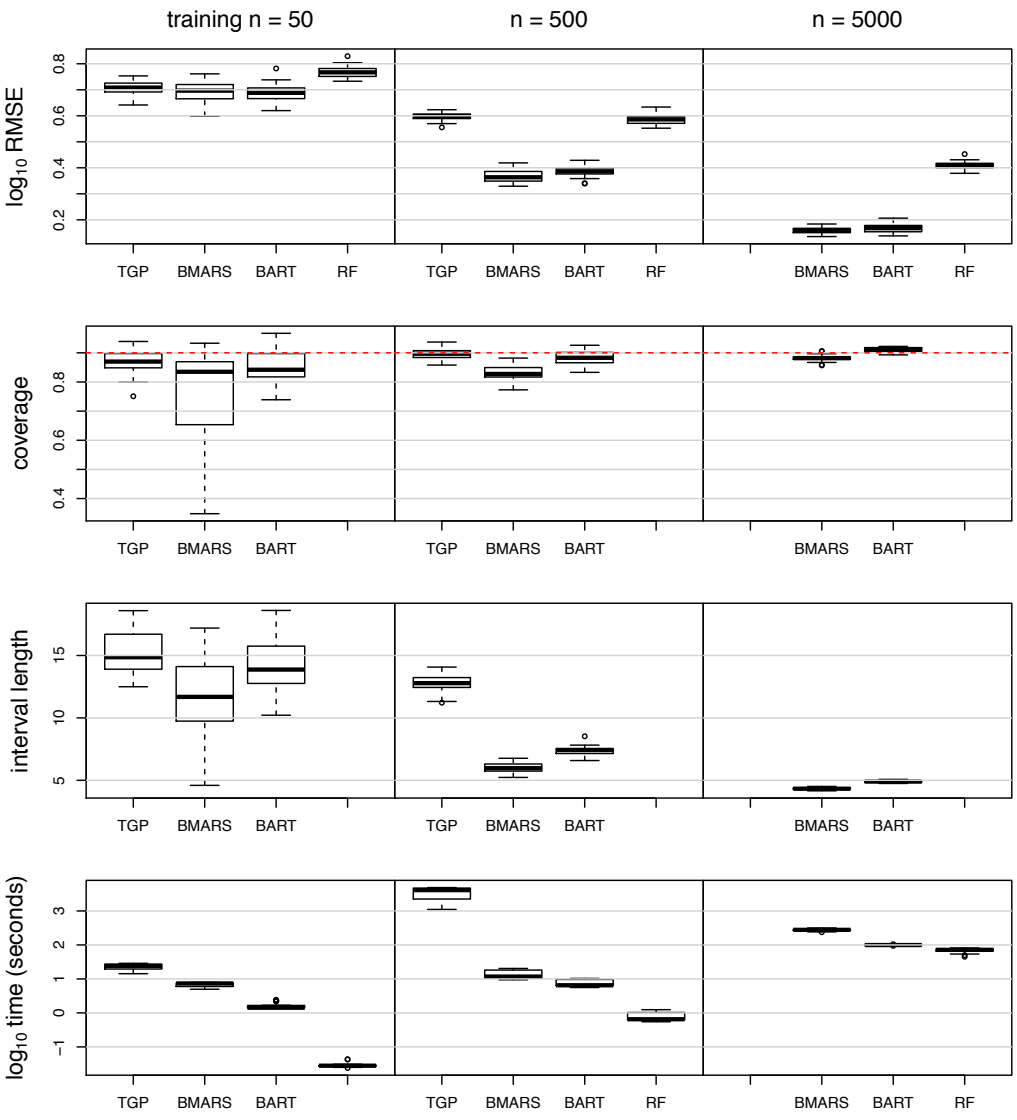


Figure 2: Same as previous figure but for EOF 1.

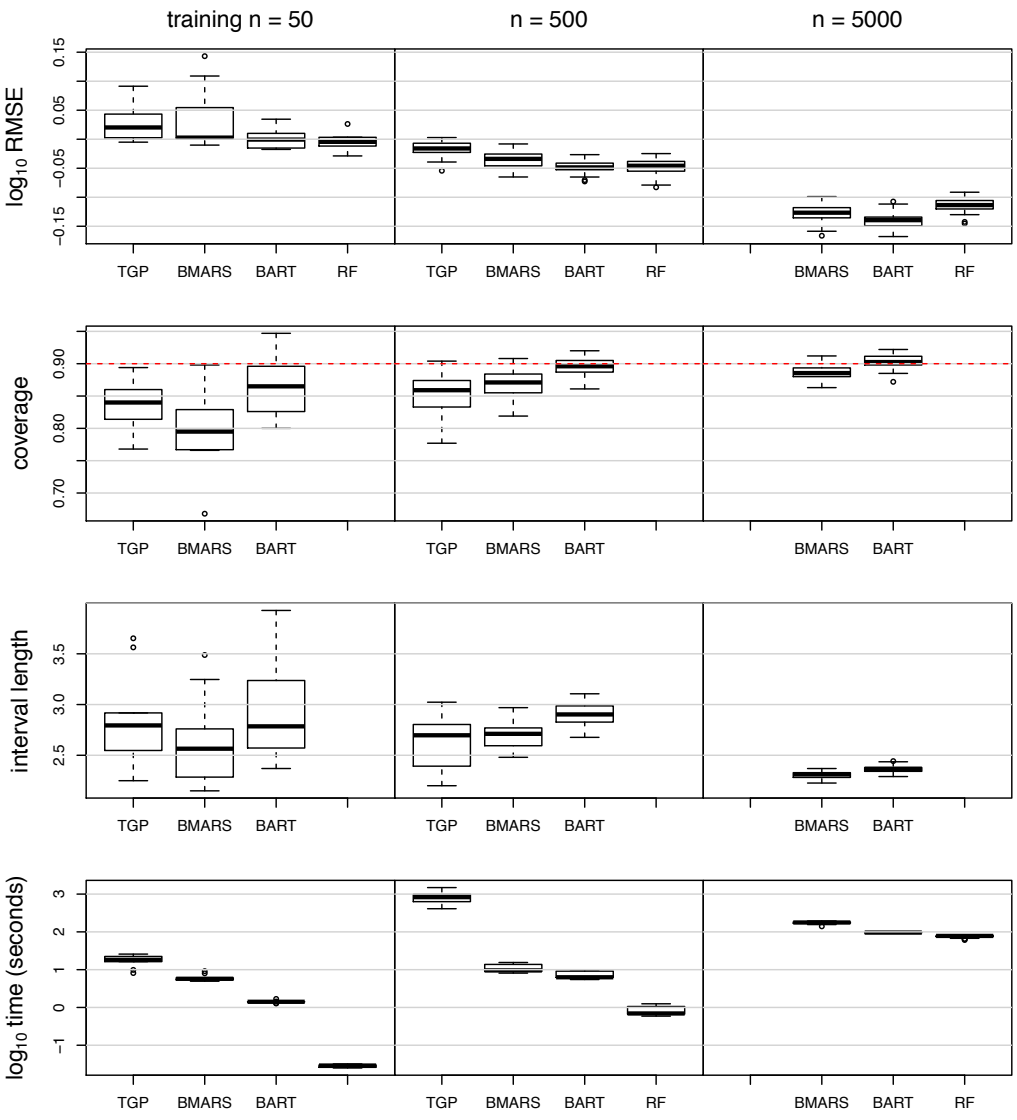


Figure 3: Same as previous figures but for EOF 50.