Statistica Sinica **28** (2018), 791-816
doi:https://doi.org/10.5705/ss.202016.0130

SENSITIVITY ANALYSIS AND EMULATION FOR FUNCTIONAL DATA USING BAYESIAN ADAPTIVE SPLINES

Devin Francom[1,2], Bruno Sansó[1], Ana Kupresanin[2]
and Gardar Johannesson[2]

[1]UC Santa Cruz and [2]Lawrence Livermore National Laboratory

Abstract: When a computer code is used to simulate a complex system, one of the fundamental tasks is to assess the sensitivity of the simulator to the different input parameters. In the case of computationally expensive simulators, this is often accomplished via a surrogate statistical model, a statistical output emulator. An effective emulator is one that provides good approximations to the computer code output for wide ranges of input values. In addition, an emulator should be able to handle large dimensional simulation output for a relevant number of inputs; it should flexibly capture heterogeneities in the variability of the response surface; it should be fast to evaluate for arbitrary combinations of input parameters, and it should provide an accurate quantification of the emulation uncertainty. In this paper we discuss the Bayesian approach to multivariate adaptive regression splines (BMARS) as an emulator for a computer model that outputs curves. We introduce modifications to traditional BMARS approaches that allow for fitting large amounts of data and allow for more efficient MCMC sampling. We emphasize the ease with which sensitivity analysis can be performed in this situation. We present a sensitivity analysis of a computer model of the deformation of a protective plate used in pressure-driven experiments. Our example serves as an illustration of the ability of BMARS emulators to fulfill all the necessities of computability, flexibility and reliable calculation on relevant measures of sensitivity.

Key words and phrases: Functional data analysis, global sensitivity analysis, multivariate adaptive regression splines, nonlinear regression, parallel tempering.

1. Introduction

Sensitivity analysis, as defined in Saltelli et al. (2004), is the study of how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input. Determining these relationships has become a fundamental step in the use of complex models because of the possible ways a modeler or model user can utilize such information.

Uses include finding which inputs require the most attention (because varying them causes the output to vary substantially), finding inputs to which the model is robust, conveying the reasons that a decision based on the model may not be totally certain, among others. More details about the practical relevance of sensitivity analysis are found in Saltelli et al. (2008) Section 1.2.14 and Pannell (1997). For the purposes of this paper, we are most interested in performing sensitivity analyses of expensive-to-run computer models that are used to simulate complex processes. Further, we would like to do this while treating the computer model as a black box, also known as non-intrusive sensitivity analysis.

While many methods for sensitivity analysis exist, they are not all of equal value. Most methods use a set of inputs at which the model needs to be evaluated, called the experimental design. Saltelli and Annoni (2010) describe the downfalls of the "one-factor-at-a-time" (OAT or OFAT) experimental design, which determines sensitivity by changing only one input at a time from some nominal input values. The OAT design is often paired with methods of sensitivity analysis deemed "local", because they only take into account variation in the model output in some small neighborhood of the inputs, usually by taking or approximating a derivative Saltelli, Tarantola and Campolongo (2000). By contrast, "global" sensitivity analysis methods (Sobol' (1990); Saltelli et al. (2008)) take into account the entire space of uncertain inputs by eliciting probability distributions over the inputs. Global methods are thus able to discover when changes in the model output are due to simultaneous changes in multiple inputs (interactions), as well as the implications of extreme, but possible, input settings. In this paper, we limit our attention to global sensitivity analysis.

When a model is expensive to evaluate, as is the case for many computer models, being able to perform a sensitivity analysis using a limited number of model evaluations is essential. Hence, much research has been done to determine which experimental designs yield the best sensitivity analyses with the least number of evaluations (Sobol' (2001); Sacks et al. (1989)). Another approach is to use a set of model evaluations to build a fast, statistical alternative to the model. Sensitivity analysis and other analyses of uncertainty can then be performed on this surrogate model, also called an emulator or metamodel. The Gaussian process has been a popular surrogate model choice (Welch et al. (1992); Kennedy and O'Hagan (2001)) because of its flexibility and simplicity, though it is sometimes avoided because of its lack of scalability to large numbers of model evaluations, large numbers of input variables, and high-dimensional output. In general, sensitivity analysis for the surrogate model will require the surrogate to be evaluated for many different combinations of the input parameters. Oftentimes, the result is prone to Monte Carlo error, depending on the number of model evaluations. This undermines the value of surrogate models that are not extremely fast to evaluate for large numbers of inputs.

In this paper, we detail the benefits of using Bayesian multivariate adaptive regression splines (BMARS) as an emulator for the purposes of sensitivity analysis. MARS and BMARS have been recently introduced to the literature on computer model analysis (Storlie et al. (2009); Chakraborty et al. (2013); Stripling et al. (2013); Maljovec et al. (2013)), as a flexible and scalable alternative to more traditional Gaussian process based methods. We emphasize, in particular, that the sensitivity analysis of a BMARS surrogate model can be performed without requiring evaluations of the surrogate, and thus without Monte Carlo error. This is the case for a limited number of surrogate models, perhaps the most popular of which is polynomial chaos (Sudret (2008)). While both BMARS and polynomial chaos use polynomial expansions that facilitate the analytical calculations of the integrals required for a global sensitivity analysis, BMARS is especially well suited for high-dimensional problems. In addition, BMARS follows an adaptive strategy to fitting the response surface, producing flexible and parsimonious emulators. For completeness, Oakley and O'Hagan (2004) propose a method for getting sensitivity indices under a scalar Gaussian process emulator that is Monte Carlo free and analytical in some cases. However, the BMARS method we introduce is well suited to much larger datasets than Gaussian process methods.

Our interest lies especially in the application of these methods to computer models that generate functional data. Such models present a particular challenge as they often produce output of massive dimension, as, for every combination of the input parameters, there are hundreds or even thousands of simulated values. The example that motivated our interest in BMARS emulators is a computer model of the deformation of a metal plate during pressure driven experiments. This model has seven inputs detailing the configuration of the plate and outputs a curve representing the profile of the plate after deformation, as shown in Figure 1. The most natural way to approach emulating a model that outputs functions, say of r, would be to think of r as though it was another input to the model. However, if we have m model evaluations that each output a function on a grid of size n, then this approach to emulation would need to be able to handle data of size mn, which can be difficult for large m or large n.

In this paper, we show that the BMARS formulation is well suited for functional output. We introduce sensitivity analysis methods for functional data,

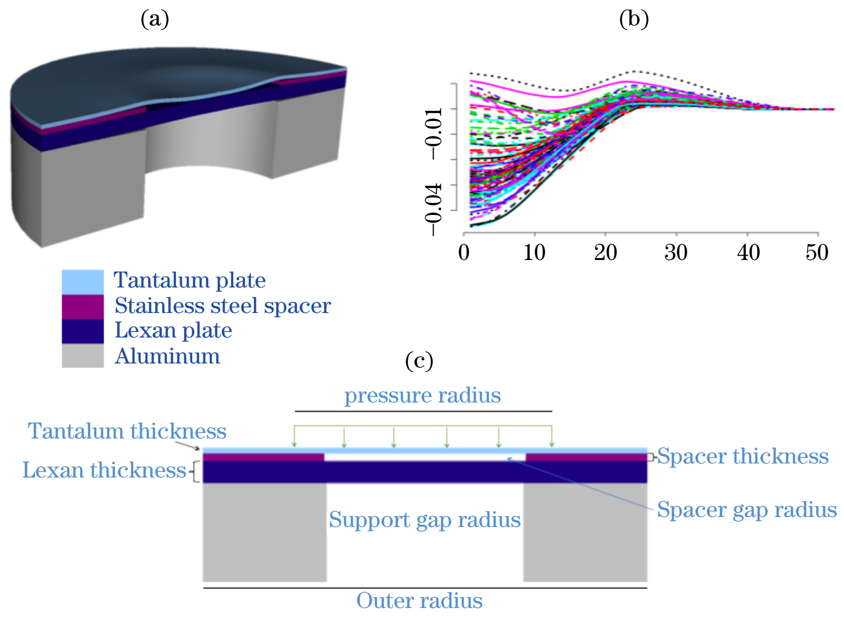

Figure 1. Formulation of the plate deformation model: (a) shows the configuration of the protective (half) plate, (b) shows the output from 104 evaluations of the computer model with different variable combinations, and (c) shows the variables. Each curve in (b) is the output from one model evaluation and represents the profile of the tantalum plate after the experiment. Zero in the x-axis of (b) represents the center of the plate.

and give analytical sensitivity measures for the functional BMARS model. We also introduce some alterations to the BMARS priors to induce regularization as well as parallel tempering in the MCMC sampling scheme to allow for efficient exploration of the highly multimodal model space.

The structure of the paper is as follows. We first review the definition of the Sobol' index, which we use as a measure of global sensitivity, in Section 2. We detail some approaches to obtaining the Sobol' index for models with functional output. We then explain our approach to fitting the BMARS surrogate model in Section 3, including our prior specification, computational approach for functional model output, tempering scheme, and analytical expressions for the Sobol' sensitivity indices. In Section 3.4, we present a simulation study to demonstrate the effectiveness of the sensitivity analysis approach. In Section 3.4, we demonstrate a sensitivity analysis of a model of the deformation of a protective plate used in pressure driven experiments. Finally, we discuss our findings in Section 3.4.

2. Global Sensitivity Indices for Functional Output

The Sobol' sensitivity indices (Sobol' (1990)) decompose the variance of the model output in terms of the variance due to main effects (first order effects) for each of the inputs, and variance due to interaction effects (higher order effects). This is the same task that the ANOVA decomposition accomplishes in linear models, but for models that may be highly nonlinear. To start, let the function f represent our model (or surrogate model). Say that f is a function of p inputs, $\mathbf{x} = (x_1, \ldots, x_p)$, each with domain in the unit interval. Further, say that f has functional output that is a function of r, also with domain in the unit interval. Then $f(r, \mathbf{x})$ is a scalar.

The Sobol' decomposition of such a functional output model could proceed in two ways. First, consider writing f as

$$f(r,\mathbf{x}) = f_0(r) + \sum_{i=1}^{p} f_i(r,x_i) + \sum_{1\le i<j\le p} f_{ij}(r,x_i,x_j) + \ldots + f_{1\ldots p}(r,x_1,\ldots,x_p)$$

where

$$\begin{aligned} f_0(r) &= \int_0^1 \cdots \int_0^1 f(r,\mathbf{x})d\mathbf{x}, \\ f_i(r,x_i) &= \int_0^1 \cdots \int_0^1 f(r,\mathbf{x})d\mathbf{x}_{-i} - f_0(r), \\ f_{ij}(r,x_i,x_j) &= \int_0^1 \cdots \int_0^1 f(r,\mathbf{x})d\mathbf{x}_{-ij} - f_0(r) - f_i(r,x_i) - f_j(r,x_j), \end{aligned}$$

and so on with $\mathbf{x}_{-ij}$ being the vector $\mathbf{x}$ without elements i and j. These terms are interpretable as the overall mean function, $f_0(r)$, the main effect function for variable i, $f_i(r, x_i)$, the two way interaction effect function for variables i and j, $f_{ij}(r, x_i, x_j)$, and so on. Proceeding, we have that

$$\begin{aligned} Var(f(r,\mathbf{x})) = &\sum_{i=1}^{p} Var(f_i(r,x_i)) + \sum_{1\le i<j\le p} Var(f_{ij}(r,x_i,x_j)) + \ldots \\ &+ Var(f_{1\ldots p}(r,x_1,\ldots,x_p)). \end{aligned} \tag{2.1}$$

Thus, we have decomposed the variance of the function in terms of the variance from the main effects of each input and the variance from the interactions between inputs. Each of these terms is a function of r, as are the associated sensitivity indices $S_{i_1\ldots i_l}(r) = Var(f_{i_1\ldots i_l}(r, i_1, \ldots, i_l))/Var(f(r,\mathbf{x}))$, the proportion of variance in the model output at r explained by the interaction between inputs $i_1 \ldots i_l$ in addition to the variance explained by main effects coming from these inputs and interactions of lesser order between these inputs. We may also be interested in the cumulative sensitivity of the model to a particular input at r. The total sensitivity for input i at r is defined as $T_i(r) = S_i(r) + \sum_{j \neq i} S_{ij}(r) + \ldots + S_{1 \ldots p}(r)$, and interpreted relative to $T_j(r)$ as the importance of input i compared to input j at r. Though these are no longer interpretable as proportions, they give us an idea of the overall importance of an input relative to the other inputs at a particular r.

The second way we could obtain the Sobol' decomposition of such a functional output model would be to augment the vector of inputs to $\mathbf{z} = (r, \mathbf{x})$. We then obtain the decomposition

$$f(z_1, \ldots, z_d) = f_0 + \sum_{i=1}^{d} f_i(z_i) + \sum_{1 \leq i < j \leq d} f_{ij}(z_i, z_j) + \ldots + f_{1 \ldots d}(z_1, \ldots, z_d), \quad (2.2)$$

where $d = p + 1$. We proceed with the traditional Sobol' variance decomposition to obtain S_i, S_{ij}, T_i, etc., which are no longer functions of r. Letting $r = z_1$, S_1 is the proportion of variance in the model output due to the main effect produced by the functional variable r. This provides interesting insights, especially when determining whether the bulk of the variance in the model output is due to the functional variables or the inputs to the computer model.

These approaches to functional sensitivity analysis have complementary strengths, as will be demonstrated in the plate deformation example in Section 3.4.

3. BMARS

We now discuss the problem of fitting a BMARS model to a set of simulated output in more detail. Multivariate adaptive regression splines (MARS) were proposed in Friedamn (1991) as a continuous alternative to recursive partitioning methods like CART. The adaptive part of MARS is what makes it work for high dimensions (large numbers of input variables). Multivariate (non-adaptive) regression splines might take a tensor product of one dimensional splines to get a multivariate spline, but with only a few dimensions the number of knots explodes and the curse of dimensionality becomes debilitating. The MARS model instead chooses knots adaptively, learning where to put them in the same way that partitions are learned in classification and regression tree (CART) models (Breiman et al. (1984)). Thus, if there is not sufficient utility in having a knot at some point in the high-dimensional input space, it will not be included. Further, the MARS model has a natural ANOVA type decomposition, making main effects and interactions easy to understand. Unlike CART, MARS produces continuous models, creating computational difficulties but resulting in more realistic models.

The original MARS inference is done using a forward stepwise algorithm followed by a backward stepwise algorithm similar to versions of inference for recursive partitioning. The Bayesian version (Denison, Mallick and Smith (1998b); Nott, Kuk and Duc (2005)) considers all the unknowns as random variables and assigns prior distributions to them. Reversible Jump Markov chain Monte Carlo (RJMCMC) (Green (1995)) methods are used to obtain transdimensional samples from the joint posterior. Our approach uses aspects of Denison, Mallick and Smith (1998b) and Nott, Kuk and Duc (2005) but with a number of significant alterations to the priors, a strong focus on efficient handling of functional data, and tempered RJMCMC to achieve more efficient sampling.

MARS is not an interpolator, so it does not have the property that the fitted emulator replicates the computer model runs exactly. When emulating deterministic computer models, using an interpolator seems like a natural choice. However, Gramacy and Lee (2012) point out that emulation is often improved when a small-scale measurement error is included.

We first formulate the model and discuss our choice of priors. We then discuss efficient computation for functional data. Next, we discuss the need for tempering, and our tempering approach. Finally, we discuss how to analytically obtain the Sobol' decomposition of a BMARS model.

3.1. Model formulation

To use the BMARS approach for functional data, we include the functional variable as an additional input to the model. At this point, we consider only the case of output as a function of one variable, as is the case in the plate deformation example. Let $y_i(r)$ denote the simulator output at r using input vector $\mathbf{x}_i$, where r denotes the functional variable and $i = 1, \ldots, n_x$. Then we model $y_i(r)$ as

$$y_i(r) = f(r, \mathbf{x}_i) + \epsilon_i(r), \quad \epsilon_i(r) \sim N(0, \sigma^2),$$

where we use a basis expansion to specify f,

$$f(r, \mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m B_m(r, \mathbf{x}).$$

If we let $\mathbf{z} = (r, \mathbf{x})$, the m^{th} basis function $B_m(r, \mathbf{x}) = B_m(\mathbf{z})$ is given by

$$B_m(\mathbf{z}) = \prod_{k=1}^{K_m} \left[s_{km} \left(z_{v_{km}} - t_{km}\right)\right]_+^{\alpha} \tag{3.1}$$

which is a tensor product of piecewise polynomials of degree α. The value of K_m determines the degree of interaction in the basis function, where $K_m \in$

$\{1, \ldots, K_{\max}\}$. The v_{km} is an index to determine which variable is used (which element of $\mathbf{z}$). We allow each variable to be used at most one time in each basis function. The t_{km} is called a knot, and is a value in the domain of the variable $z_{v_{km}}$. Following previous MARS implementations, a knot t_{km} is only allowed at one of the marginal locations where we have a value of $z_{v_{km}}$ in the data, for identifiability purposes. The s_{km} is a value in $\{-1, 1\}$. The function $[\cdot]_+$ is defined as $\max(\cdot, 0)$, meaning that it makes any negative values zero.

This notation follows that of Friedamn (1991), with the exception of the inclusion of the functional variable. The functional variable could be space, time, or any other such variable over which the output is measured. In practice, functional output is usually given on a grid. While nothing in the formulation above requires the output for $y_i(r)$ to be on the same grid of r values as $y_j(r)$, for simplicity we assume that this is the case: $y_1(r), \ldots, y_n(r)$ are given on the same grid of n_r values. We denote this grid as $\mathbf{r}$. For example, if r denotes time, n_r would be the number of time points on which the output is given, and $\mathbf{r}$ would be the vector of those time points. Treating the functional variable as an additional input results in a univariate MARS fitted over $N = n_x \times n_r$ data points. If we define the $n_x \times p$ matrix $\mathbf{X}$ such that the i^{th} row of $\mathbf{X}$ is $\mathbf{x}_i$, then the data used to fit the MARS model are the rows of $[\mathbf{1}_{n_x} \otimes \mathbf{r}, \mathbf{X} \otimes \mathbf{1}_{n_r}]$, where $\otimes$ denotes the Kronecker product. For large n_x or n_r manipulating, and even storing the full matrix $\mathbf{X}$ can be challenging. More specifically, in the course of inference and prediction, we plug in values of $\mathbf{x}$ and r to (3.1) to get discretized versions of the basis functions, which we call basis vectors. We denote the m^{th} basis vector as $\mathbf{B}_m$, and the $(M+1) \times N$ matrix of basis vectors (with an additional column of ones) as $\mathbf{B}$. When N is large and M is moderately large, storing this matrix is costly. We will discuss a computational strategy that simplifies this approach in Section 3.2.

To complete the Bayesian specification of the model, we specify priors for the unknown parameters. Our unknowns include the number of basis functions M, the basis function coefficients $\mathbf{a} = (a_1, \ldots, a_M)$, the variance σ^2, and the parameters used to build each basis function. For the m^{th} basis function, these are K_m, $\mathbf{s}_m = (s_{1m}, \ldots, s_{K_m m})$, $\mathbf{v}_m = (v_{1m}, \ldots, v_{K_m m})$, and $\mathbf{t}_m = (t_{1m}, \ldots, t_{K_m m})$. As notation, let $\mathbf{K} = (K_1, \ldots, K_M)$, $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_M)$, $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_M)$, and $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_M)$. We formulate our prior as

$$p\left(M,\sigma^2,\mathbf{a},\mathbf{K},\mathbf{s},\mathbf{v},\mathbf{t},\lambda,\tau\middle|\,\mathbf{X},\mathbf{r}\right)=p(\lambda)p(\tau|\mathbf{X},\mathbf{r})p(\sigma^2)p(M|\lambda)p(\mathbf{a}|M,\sigma^2,\mathbf{B},\tau)\prod_{m=1}^{M}p(K_m|M)p(\mathbf{s}_m,\mathbf{v}_m,\mathbf{t}_m|K_m,M,\mathbf{X},\mathbf{r})$$

where λ and τ are hyper parameters to be discussed below. We point out that $\mathbf{B}$ is a function of M, $\mathbf{K}$, $\mathbf{s}$, $\mathbf{v}$, $\mathbf{t}$, $\mathbf{X}$, and $\mathbf{r}$. Priors conditional on such quantities as $\mathbf{X}$ and $\mathbf{r}$ do not violate the Bayesian formulation, as they are considered known.

For the number of basis functions we use a Poisson prior $p(M|\lambda)\propto e^{-\lambda}\lambda^M/M!$ truncated to $M=0,\ldots,M_{\max}$, where $M_{\max}$ is the maximum allowable number of basis functions. We use a $Gamma(a_\lambda,b_\lambda)$ hyperprior for λ. For the error variance, we use a default prior, $p(\sigma^2)\propto 1/\sigma^2$. For the basis function coefficients, we use a variant of Zellner's g prior (Zellner (1986); Liang et al. (2008)) with $\mathbf{a}|\mathbf{B},\tau,\sigma^2\sim N(\mathbf{0},\sigma^2/\tau\,(\mathbf{B}'\mathbf{B})^{-1})$ where $\tau|\mathbf{X},\mathbf{r}\sim Gamma(1/N,1)$. Thus, τ is centered over the unit information prior. This prior simplifies computations when compared to previous approaches that use a ridge regression prior while still inducing regularization by introducing shrinkage. For the interaction order we use a discrete uniform prior $K_m|M\sim Unif\{1,\ldots,K_{\max}\},\ m=1,\ldots,M$. Most implementations of this model take $K_{\max}=2$ as a default. However, we would like to allow for two-way interactions between the input variables that can also interact with the functional variable, so we use $K_{\max}=3$. One could easily adopt a prior that assigns decreasing probability to larger values of K in order to allow for higher dimensional interactions if the data really dictate their inclusion.

For the signs, variables, and knots, previous Bayesian approaches have considered a discrete uniform prior over all possibilities. We use a similar discrete uniform prior, but over a limited set of possibilities. We want to limit how localized a basis function can become in the same way that recursive partitioning approaches do, which is to require that each partition contain a certain number of data points. Very local fitting often produces overfitting and, in our experience, functional output magnifies this issue. In the MARS approach, we do not exactly have regular partitions, since basis functions are usually overlapping in the input space. We get a gauge on how local the structure a certain basis function is trying to explain by counting the number of non-zero points in the basis vector. Although this is only a reliable measure of how localized the model is near the edge of the input space (in p dimensions), this is the part of the space for which MARS tends to become unstable.

As an example of the instabilities that we can encounter if we allow for all possible knot, sign, and variable combinations, consider the dataset given in

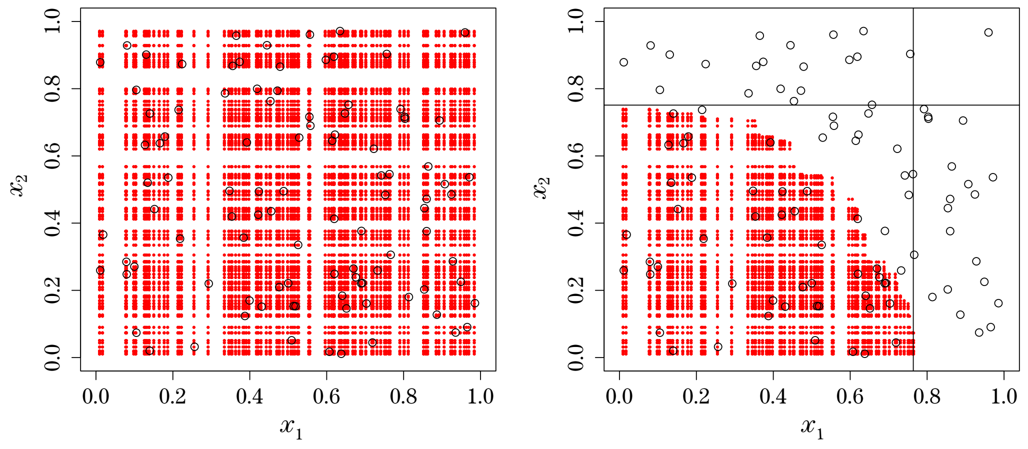

Figure 2. In both plots, 100 random uniform (x_1, x_2) pairs are shown with large circles. If we wanted to build the basis function $[x_1 - t_1]_+[x_2 - t_2]_+$, the small dots in the left panel show the possible (t_1, t_2) knot locations if the prior for knots is unconstrained. On the right are the possible knot locations when the resulting basis vector is constrained to have at least 20 non-zero values. The lines are placed so that there are 20 data points larger than them in each dimension marginally to illustrate the approach advocated by Friedman.

Figure 2. There are 100 random uniform (x_1, x_2) pairs, given as large circles. If we were interested in choosing knots for the MARS basis function $[x_1-t_1]_+[x_2-t_2]_+$, the small dots in the left plot give the possible locations if we do not constrain the prior. The partition created by the choice of (t_1, t_2) would, in many of these cases, contain few data points. The corresponding basis function would then be trying to fit the very local structure in those few points. The right panel shows the possible knot locations (small dots) if we require the partition to contain at least 20 points. It may appear that a simpler way to fix the edge instability of MARS would be to not allow marginal knots too close to the endpoints of the space, as suggested in Friedamn (1991). For instance, if we required each knot in the example to have at least 20 points marginally between it and the edge of the space, we would allow for a knot at the intersection of the lines shown in the right panel. However, there would only be one data point driving the fit of that basis function, which could lead to overfitting.

To specify the prior for the signs, variables, and knots, that correspond to our proposed constraint, we use the discrete uniform distribution

$$p(\mathbf{s}_m, \mathbf{v}_m, \mathbf{t}_m | K_m, M, \mathbf{X}, \mathbf{r}) = \begin{cases} c_{K_m} & \text{if } b_m \geq b, \\ 0 & \text{otherwise,} \end{cases}$$

where b_m is the number of non-zero values in the basis vectors and b is the minimum number of non-zero points. In practice, since we have the entire functional output for each input combination, we might consider choosing b based only on the part of the basis function that corresponds to the non-functional inputs. We can do this by replacing b above with bn_r where b is chosen as the minimum number of input points allowed to contribute to the local structure of the function, which we do in the plate deformation problem. In particular, we use $b = 20$, though we find the results are fairly robust for $b > 10$. As with the prior for K, we could instead use a prior that places lower probability on basis functions with smaller b_m to allow for more localized basis functions if the data gave strong enough evidence for them, though we consider only the discrete uniform prior in this paper. The value of c_{Km} would usually be unimportant and this prior would merely add an indicator function to the posterior. However, in the RJMCMC algorithm, c_{Km} will play a role. The actual value of c_{Km} is the reciprocal of the number of possible basis functions with interaction order K_m, which depends on $\mathbf{X}$ and $\mathbf{r}$. For some datasets, it is feasible to run a pre-processing step to count the basis functions that meet the required criteria. For datasets where this is not the case, we recommend using, as a conservative proxy, the constant that would result from the unconstrained prior as an estimate,

$$c_{Km} = \left(\frac{1}{2}\right)^{K_m} \binom{p}{K_m}^{-1} \prod_{k=1}^{K_m} \frac{1}{n_{v_{km}}},$$

where the first term is obtained from the count of all $\mathbf{s}_m$, the second from that of the $\mathbf{v}_m$, and the third from that of the $\mathbf{t}_m$, conditional on the corresponding $\mathbf{v}_m$. Here, $n_{v_{km}}$ is the number of unique marginal values of the v_{km} input in the dataset. This prior for the knots is slightly different than previous approaches because we have n_x possible knot locations if v_{km} is one of the regular inputs, and n_r possible knot locations if v_{km} is the functional variable. We find that using this estimate does not negatively influence the results in test cases.

Now, to address our settings of the remaining parameters, we first consider the spline order α. While in many cases, setting α to an integer that would ensure continuous derivatives would make sense, we often encounter instabilities when $\alpha > 1$ because of the erratic tail behavior of higher order polynomials (Friedamn (1991)). In practice, $\alpha = 1$ tends to work quite well even for smooth surfaces. We find that the value of $M_{\max}$ need not be finite unless computer memory constraints are encountered in the fitting. $M_{\max}$ too small should be avoided in order to allow the necessary exploration of the parameter space. The setting of the hyperparameters for λ, the mean of the distribution of the number of basis functions, can be difficult. Under the model formulation we have given, the only way to prevent overfitting, even with a reasonable setting of b and $K_{\max}$, is to have a strong prior keeping the number of basis functions small. Hence, this is a prior that may require tuning. Other possibilities are to use a strong prior keeping the value of σ^2 large or, as has been done in other approaches, to use a large value of τ to shrink the values of $\mathbf{a}$ towards zero. In our experience, changing the value of τ has little effect. A prior keeping the value of M small seems less invasive than one that keeps σ^2 large, especially when uncertainty quantification is important. Thus, we advocate for a prior that keeps λ smaller than the actual number of basis functions that we expect. Otherwise, the slow rate at which the tail of the Poisson distribution decays makes it a weak prior.

3.2. Computation

We adapt the original RJMCMC stochastic search algorithm for BMARS (Denison, Mallick and Smith (1998b) with the additions of Nott, Kuk and Duc (2005)) to work with the priors discussed in the previous section. The RJMCMC has three possible steps: birth (create a new basis function, add it to the model), death (remove a basis function), change (change a knot and sign in one basis function). The additions of Nott, Kuk and Duc (2005) allow for fast variable selection by proposing variables to be used in a new basis function with probability proportional to the number of times they have been used in the current set of basis functions.

To efficiently deal with large N, we break each of our tensor product basis functions into two parts: one that uses the variables $\mathbf{x}$ and one that uses the functional variable r. We can then write basis function m as $B_m(r,\mathbf{x}) = B_m^x(\mathbf{x})B_m^r(r)$. If we let $\mathbf{S}_m^x = \{k \in (1,\ldots,K_m) : z_{km} \neq r\}$ and $\mathbf{S}_m^r = \{k \in (1,\ldots,K_m) : z_{km} = r\}$, then

$$B_m^x(\mathbf{x}) = \begin{cases} \prod_{k\in\mathbf{S}_m^x}[s_{km}(z_{v_{km}} - t_{km})]_+^\alpha & \text{if } |\mathbf{S}_m^x| > 0, \\ 1 & \text{otherwise,} \end{cases}$$

$$B_m^r(r) = \begin{cases} [s_{km}(r - t_{km})]_+^\alpha|_{k\in\mathbf{S}_m^r} & \text{if } |\mathbf{S}_m^r| > 0, \\ 1 & \text{otherwise.} \end{cases}$$

We then create the n_x-vector $\mathbf{B}_m^x = (B_m^x(\mathbf{x}_1),\ldots,B_m^x(\mathbf{x}_{n_x}))'$ and the n_r-vector $\mathbf{B}_m^r = (B_m^r(r_1),\ldots,B_m^r(r_{n_r}))'$. Then a MARS basis vector is written as $\mathbf{B}_m = \mathbf{B}_m^x \otimes \mathbf{B}_m^r$. Thus, we have broken the MARS basis vector into the part from the input variables ($\mathbf{B}_m^x$) and that from the functional variable ($\mathbf{B}_m^r$). The cases when these are vectors of ones occur when the basis function uses only the functional variable or only the design variables.

The matrix of basis functions $\mathbf{B}$ can be written as $[\mathbf{1}_N, \mathbf{B}_1^x \otimes \mathbf{B}_1^r, \ldots, \mathbf{B}_M^x \otimes \mathbf{B}_M^r]$. If $\mathbf{B}^x = [\mathbf{1}_{n_x}, \mathbf{B}_1^x, \ldots, \mathbf{B}_M^x]$ and $\mathbf{B}^r = [\mathbf{1}_{n_r}, \mathbf{B}_1^r, \ldots, \mathbf{B}_M^r]$, then we have $\mathbf{B} = \mathbf{B}^x * \mathbf{B}^r$ where $*$ denotes the Khatri-Rao product (Kolda (2006); Lev-Ari et al. (2005)). This Kronecker structure simplifies many matrix calculations. For instance, from properties of Khatri-Rao products, $\mathbf{Ba} = \text{vec}(\mathbf{B}^r \text{diag}(\mathbf{a}) \mathbf{B}^{x\prime})$. This is an important quantity in prediction, and under this formulation it can be calculated without explicitly building the whole matrix $\mathbf{B}$. Other important quantities also have simplified forms such as $\mathbf{B}'\mathbf{B} = (\mathbf{B}^{x\prime}\mathbf{B}^x) \circ (\mathbf{B}^{r\prime}\mathbf{B}^r)$, where $\circ$ denotes the Hadamard (elementwise) product, and $\mathbf{B}'\text{vec}(\mathbf{Y}) = \text{vecd}(\mathbf{B}^{r\prime}\mathbf{Y}\mathbf{B}^x)$, where $\mathbf{Y}$ is the $n_r \times n_x$ matrix of data. Here, $\text{vec}(\cdot)$ denotes the columnwise stacking of a matrix and $\text{vecd}(\cdot)$ denotes the vectorization of the diagonal of a matrix. These simplifications mean that we do not need to build $\mathbf{B}$ explicitly, but can instead work with the smaller $\mathbf{B}^r$ and $\mathbf{B}^x$.

Further simplifications come from the fact that in the RJMCMC algorithm, we are only adding, deleting, or changing one basis function at a time. For instance, if we were adding a basis function, $\mathbf{B}_*$, we need only calculate $\mathbf{B}_*{}'\mathbf{B}_* = (\mathbf{B}_*^{x\prime}\mathbf{B}_*^x)(\mathbf{B}_*^{r\prime}\mathbf{B}_*^r)$ and $\mathbf{B}'\mathbf{B}_* = (\mathbf{B}^{x\prime}\mathbf{B}_*^x) \circ (\mathbf{B}^{r\prime}\mathbf{B}_*^r)$ in order to update $\mathbf{B}'\mathbf{B}$. Deleting is simpler in that if we have selected the m^{th} basis function to delete, we need only remove the $(m+1)^{\text{th}}$ row and column of $\mathbf{B}'\mathbf{B}$. We use similar updates for $\mathbf{B}'\text{vec}(\mathbf{Y})$.

3.3. Tempering

The BMARS model, along with Bayesian versions of CART (Denison, Mallick and Smith (1998a); Chipman, George and McCulloch (1998)), yields a posterior distribution over a space of models that is often highly multimodal. As such, the RJMCMC algorithm tends to have difficulty exploring the entire posterior. In the BMARS case, after selecting a set of basis functions and ending up in one of these modes, it is unlikely that enough basis functions will be deleted to allow for the chain to move to another mode. BMARS and Bayesian CART approaches to dealing with this problem have often centered around restarting the MCMC algorithm after a certain number of iterations. Restarting the MCMC, or equivalently, taking samples from parallel chains, has the undesirable property of representing modes based on their "basins of attraction" instead of the total probability associated with each mode (Neal (1996)).

We overcome these problems by using parallel tempering, also known as Metropolis coupled MCMC (Geyer (1991)). We run multiple MCMC chains in parallel, each with a slightly different stationary distribution, and allow them to swap states. Particularly, if we denote our parameter vector as $\boldsymbol{\theta} = (M, \sigma^2, \mathbf{a}, \mathbf{K}, \mathbf{s}, \mathbf{v}, \mathbf{t}, \lambda, \tau)$, our data as $\mathbf{y}$, and the posterior of interest as $\pi(\boldsymbol{\theta}|\mathbf{y})$, we define a series of altered posterior distributions as $\pi_1(\boldsymbol{\theta}|\mathbf{y}), \ldots, \pi_T(\boldsymbol{\theta}|\mathbf{y})$. Here, we define $\pi_i(\boldsymbol{\theta}|\mathbf{y}) \propto \pi(\boldsymbol{\theta}|\mathbf{y})^{t_i}$, where t_i is called the inverse temperature parameter and the sequence $1 = t_1 > t_2 > \cdots > t_T > 0$ is called the temperature ladder. Small values of t_i flatten posterior modes and raise troughs, and correspond to "heated" chains where mixing is easier. A swap of the current state from chain i (called $\boldsymbol{\theta}_i$) with that of the current state from chain j (called $\boldsymbol{\theta}_j$) is accepted with probability

$$\alpha_{\text{swap}} = \min\left\{1, \frac{\pi_i(\boldsymbol{\theta}_j|\mathbf{y})\pi_j(\boldsymbol{\theta}_i|\mathbf{y})}{\pi_i(\boldsymbol{\theta}_i|\mathbf{y})\pi_j(\boldsymbol{\theta}_j|\mathbf{y})}\right\}.$$

The state vectors $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are possibly of different dimension, so quantities like $\pi_i(\boldsymbol{\theta}_j|\mathbf{y})$ are not necessarily intuitive. Further, the assurance that posterior normalizing constants cancel is not immediately clear, because of the dimensionality difference. We will show explicitly what we mean when we write $\pi_i(\boldsymbol{\theta}_j|\mathbf{y})$ and why it works with the unnormalized posterior.

Call the unnormalized posterior distribution function $g(\boldsymbol{\theta}|\mathbf{y})$. Then, $\pi(\boldsymbol{\theta}|\mathbf{y}) = g(\boldsymbol{\theta}|\mathbf{y})/c$ where c is obtained by marginalizing $g(\boldsymbol{\theta}|\mathbf{y})$ over all the parameters in $\boldsymbol{\theta}$. Note that c is the same no matter the dimension of $\boldsymbol{\theta}$ since we are marginalizing over all possible dimensions. What we mean by $\pi_i(\boldsymbol{\theta}|\mathbf{y})$, no matter the dimension of $\boldsymbol{\theta}$, is $\pi_i(\boldsymbol{\theta}|\mathbf{y}) = [g(\boldsymbol{\theta}|\mathbf{y})/c]^{t_i}/a_i$ where a_i is obtained from marginalizing $[g(\boldsymbol{\theta}|\mathbf{y})/c]^{t_i}$ over all the parameters in $\boldsymbol{\theta}$. If $c_i = c^{t_i}a_i$, we have that $\pi_i(\boldsymbol{\theta}|\mathbf{y}) = g(\boldsymbol{\theta}|\mathbf{y})^{t_i}/c_i$, and then $\pi_i(\boldsymbol{\theta}_j|\mathbf{y})$ and $\pi_i(\boldsymbol{\theta}_i|\mathbf{y})$ have the same normalizing constants, c_i. Thus, the acceptance ratio can be written as

$$\frac{\pi_i(\boldsymbol{\theta}_j|\mathbf{y})\pi_j(\boldsymbol{\theta}_i|\mathbf{y})}{\pi_i(\boldsymbol{\theta}_i|\mathbf{y})\pi_j(\boldsymbol{\theta}_j|\mathbf{y})} = \left(\frac{g(\boldsymbol{\theta}_j|\mathbf{y})}{g(\boldsymbol{\theta}_i|\mathbf{y})}\right)^{t_i - t_j}$$

which means that the normalizing constants cancel and we need only keep track of the unnormalized posterior for each chain.

This can be done in parallel, as in Altekar et al. (2004), so that it requires very little communication between chains. Rather than explicitly swapping states, we swap temperatures. We only keep samples from the true posterior (the coolest chain), so the memory footprint of such an algorithm is not substantially larger than when we do not use tempering. While simulated tempering

(Geyer and Thompson (1995)) might provide better mixing than parallel tempering (Atchadé, Roberts and Rosenthal (2011)), specification of a pseudo-prior for the inverse temperature is difficult. Further, being able to run the chains in parallel is very useful to those who have access to the large parallel computers that are common among those working in uncertainty quantification.

3.4. Sobol' indices for BMARS

We now discuss the Sobol' decomposition of a BMARS model. In particular, we note that the required integration can be done in closed form. Chen, Jin and Sudjianto (2005) describe conditions under which tensor product basis function expansions have analytical Sobol' decompositions. For MARS models, these conditions are that we can analytically perform the integration

$$C_v(s,t) = \int_0^1 [s(x_v - t)]_+^\alpha dx_v, \tag{3.2}$$

$$C_v(s_1,t_1,s_2,t_2) = \int_0^1 [s_1(x_v - t_1)]_+^\alpha [s_2(x_v - t_2)]_+^\alpha dx_v. \tag{3.3}$$

We find that this is possible when α is a positive integer. Particularly,

$$C_v(s,t) = \begin{cases} \dfrac{(1-t)^{\alpha+1}}{\alpha+1} & s=1, \\ \dfrac{t^{\alpha+1}}{\alpha+1} & s=-1, \end{cases}$$

$$C_v(s_1,t_1,s_2,t_2) = \begin{cases} \int_{t_2}^1 [(x_v-t_1)(x_v-t_2)]^\alpha \, dx_v & s_1=s_2=1, \\ \int_0^{t_1} [(x_v-t_1)(x_v-t_2)]^\alpha \, dx_v & s_1=s_2=-1, \\ (-1)^\alpha \int_{t_1}^{t_2} [(x_v-t_1)(x_v-t_2)]^\alpha \, dx_v & s_1=1, s_2=-1, \\ 0 & s_1=-1, s_2=1, \end{cases} \tag{3.4}$$

assuming, without loss of generality, that $t_1 \le t_2$. The integrals in (3.4) are

$$\int_a^b [(x-t_1)(x-t_2)]^\alpha \, dx = \left[\sum_{i=0}^{\alpha} p_i (x-t_1)^{\alpha-i} (x-t_2)^{\alpha+1+i} \right]_{x=a}^{x=b},$$

where $p_i = ((\alpha!)^2(-1)^i)/((\alpha-i)!(\alpha+1+i)!)$. We can then write quantities like $\hat{f}_{ij}(z_i,z_j) = \int_0^1 \cdots \int_0^1 f(\mathbf{z}) d\mathbf{z}_{-ij}$, the non-centered version of $f_{ij}(z_i,z_j)$ in (2.2), using (3.2). We can write quantities like $\int_0^1 \int_0^1 \hat{f}_{ij}(z_i,z_j)^2 dz_i dz_j$, important in finding $Var(f_{ij}(z_i,z_j))$, using (3.2) and (3.3). Similarly, $f_{ij}(r,x_1,x_2)$ and $Var(f_{ij}(r,x_1,x_2))$ can be written using (3.2) and (3.3) where we never integrate over r.

Specifically, if we are considering a set of variables indexed by $W = \{i_1, \ldots,$

$i_l\}$, the quantity of interest from (2.1) can be obtained as

$$Var(f_W(r,\mathbf{x}_W)) = \sum_{U\in P}(-1)^{|W|-|U|}V_U(r),$$

where P is the power set of W, excluding the empty set, $|U|$ denotes the size of set U, and

$$V_U(r) = \sum_{m_1=1}^{M}\sum_{m_2=1}^{M} a_{m_1}a_{m_2}B^r_{m_1}(r)B^r_{m_2}(r)\left\{\prod_{k\in U_1} C_{v_{km_1}}(s_{km_1},t_{km_1}) \prod_{k\in U_2} C_{v_{km_2}}(s_{km_2},t_{km_2}) \prod_{k\in U_{12}} C_{v_{km_1}}(s_{km_1},t_{km_1},s_{km_2},t_{km_2}) - \prod_{k\in \mathbf{S}_x} C_{v_{km_1}}(s_{km_1},t_{km_1}) \prod_{k\in \mathbf{S}_x} C_{v_{km_2}}(s_{km_2},t_{km_2})\right\}.$$

Here $U_1 = \{k : \forall l, v_{km_1} \neq v_{lm_2}\}$, U_2 is defined similarly with m_1 and m_2 switched, $U_{12} = \{k : v_{km_1} \in U\} \cap \{k : v_{km_2} \in U\}$, and $B^r_m(r)$ and $\mathbf{S}_x$ are defined as in Section 3.2.

For each posterior sample, we can calculate the Sobol' decomposition to quantify the uncertainty of the sensitivity indices. This would be uncertainty due to variance in the emulator, not due to Monte Carlo error, as is usually the case.

4. Simulation

Consider the function $f(\mathbf{x}) = 10\sin(2\pi x_1x_2) + 20\,(x_3-0.5)^2 + 10x_4 + 5x_5$, with each $x_i \in [0,1]$, a slight alteration of the Friedman function (Friedamn (1991); Denison, Mallick and Smith (1998b); Gramacy and Lee (2012); Surjanovic and Bingham (2015)). We treat x_1 as a functional variable. In order to get more flexible functional output we replace π in the Friedman function with 2π. The integrals necessary to obtain the Sobol' sensitivity indices for this function are mostly analytical, with derivations given in the supplementary material. The integrals that are not analytical have solutions that can be represented using series approximations to an arbitrary degree of accuracy.

We simulated n_x values of $x_2,\ldots,x_5$ from the uniform hypercube and set x_1 to be a grid of values of length n_r and used these to generate $f(\mathbf{x})$ for the $n_x \times n_r$ combinations of $\mathbf{x}$. We added standard Normal errors to the simulated values of $f(\mathbf{x})$. We further generated n_x values of $x_6,\ldots,x_p$ as extraneous variables. We fit our BMARS emulator to these data. This corresponds to having n_x model runs, each of which outputs a curve on a grid of n_r values. Further, the computer

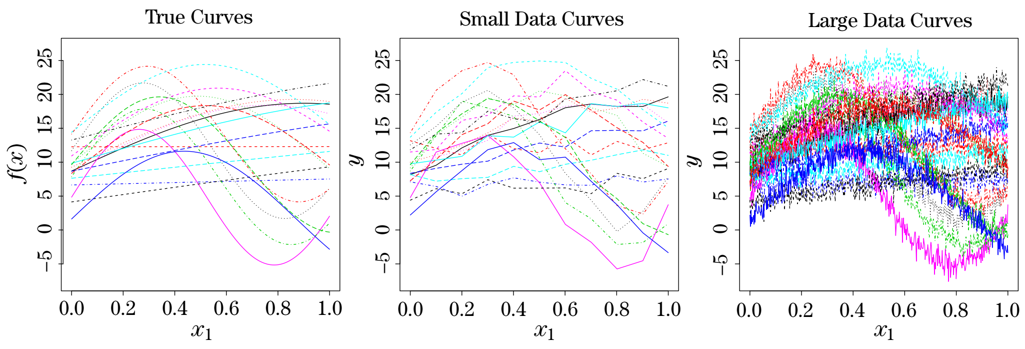

Figure 3. A few simulated curves, $f(\mathbf{x})$, are given on the left in terms of the functional variable x_1. The middle and right plots show the small and large data curves corresponding to the curves on the left.

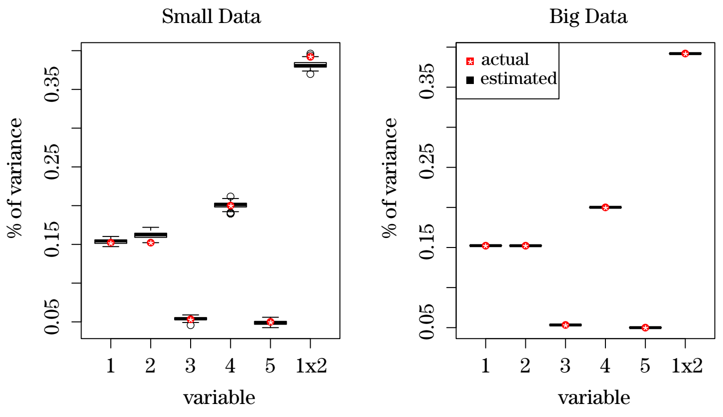

Figure 4. Posterior distributions of sensitivity indices represented with boxplots for the small and large datasets. Actual values of sensitivity indices are also shown.

model takes p inputs, but only four of them are meaningful.

We considered two settings of n_x, n_r, and p that demonstrate cases with small and large data. For the small data case, we used $n_x = 100$, $n_r = 10$, and $p = 5$. For the large data case, we used $n_x = 20{,}000$, $n_r = 500$, and $p = 200$. Twenty curves with different settings of $\mathbf{x}$ are shown in Figure 3 under these two settings of n_r with standard Normal error, in addition to the true curves. Fitting BMARS models to the small and large datasets took 14 and 966 seconds, respectively, on a personal computer with a 1.7 GHz Intel Core i7 processor. The posterior median number of basis functions used for the small and large datasets was 26 and 95. In these simulations, we did not use tempering. We used default values of priors for the small data case. In the large data case,

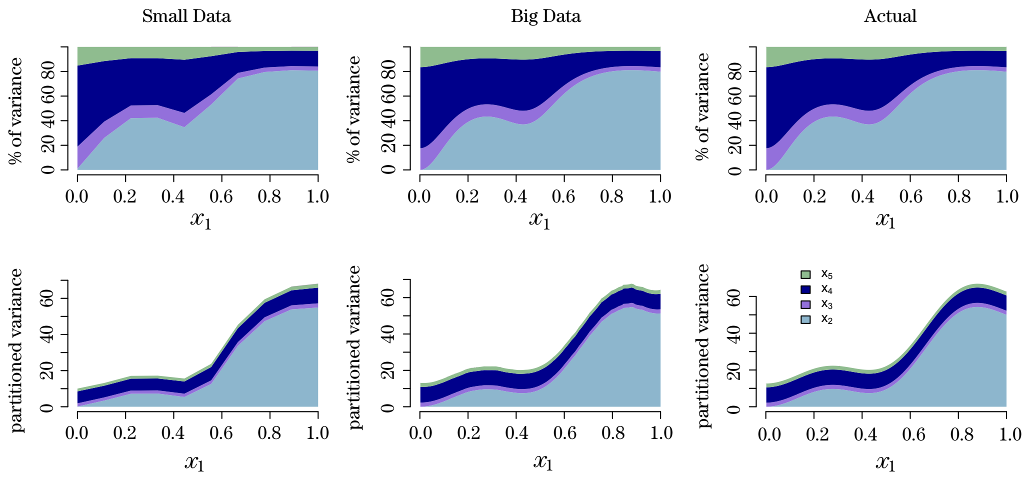

Figure 5. Functional pie charts of posterior mean sensitivity indices compared to true values. The left panels show the sensitivity indices using the smaller data set. The middle shows these indices using the larger dataset. The right panels show true values. The top panels show functional pie charts while the bottom show the partitioned variance.

we used a strong prior on M to keep the number of basis functions small for computational purposes.

We demonstrate our two functional sensitivity analysis approaches using BMARS emulators built using these datasets. The sensitivity indices when x_1 was treated as one of inputs are given in Figure 4, along with the true values. The functional sensitivity indices (posterior mean) are given in the form of functional pie charts (Saltelli, Tarantola and Campolongo (2000); Lamboni et al. (2009)) in Figure 5, along with the true functional sensitivity indices. In both approaches, we were able to capture the important elements quite well for both the small and large data cases. Since the Sobol' indices are available analytically when we use the BMARS emulator, discrepancy in the sensitivity indices (which was quite small in our simulation) is due only to emulation discrepancy. The sensitivity analysis was more accurate when we used more data and had more uncertainty when we have small data, as we would expect. Our BMARS methods were able to efficiently handle the large data analysis, where use of the Gaussian process would have been infeasible.

Deterministic computer model output would not have unexplained noise like our simulation does. Similar simulations with noiseless data produce similar sensitivity indices, though regularization becomes more important when fitting the BMARS model to limit the number of basis functions used. This limitation prevents overfitting in the small data case. While the large data case has no issues with overfitting, limiting the number of basis functions is desirable for computational purposes.

5. Plate Deformation Model Sensitivity Analysis

We used the above formulation to create an emulator for the plate deformation model and performed a sensitivity analysis. The plate deformation model has seven inputs controlling the configuration of a tantalum plate used to protect a diagnostic imager during pressure-driven experiments, as shown in Figure 1(c). The output from one model run is a curve representing the profile of the deformed plate starting from the center of the plate and extending along the radius to the end of the plate. Hence, the functional variable r is called radial position. Each curve is given on the same grid of $n_r = 517$ equally spaced points. We had model runs corresponding to $n_x = 104$ combinations of $\mathbf{x}$ specified using a Latin hypercube design, for a total of 53,768 simulated values. A separate Latin hypercube was used to obtain 34 model runs that were used to test the fit of the emulator.

We used the BMARS formulation given above to build a surrogate model. We used $\alpha = 1$, $M_{\max} = 300$, and a hyperprior for λ that kept it close to zero. Specifically, we used $a_\lambda = 1$ and $b_\lambda = 10^{300}$ (chosen by cross-validation). Such a large value of b_λ was necessary in this case because of the combination of the smoothness of the curves and the large number of possible basis functions. Had we not limited basis functions based on partition size, there would have been more than 1.2×10^9 possible basis functions, which means more than $2^{1.2\times 10^9}$ possible models. Here, $b_\lambda = 10^{300}$ controls the regularization on the number of basis functions and is not an exorbitantly large number when compared with the size of the model space. The resultant Gamma hyperprior has so little variance that it may be unnecessary here, but we included it for consistency. When cross-validating, we compared mean squared error averaged over radial position for values of b_λ on the $\log_{10}$ scale from 0 (results in the maximum number of basis functions) to 305. We considered other values of the maximum degree of interaction (fixed at $K_{\max} = 3$) and the minimum number of non-zero values in a basis function (fixed at $b = 20$), but did not find better cross-validated fits.

When the curves are smooth, σ^2 plays the role of quantifying the variance in the data not explained by the model, and will thus be larger as we impose stronger regularization. Posterior exploration is especially difficult in this case

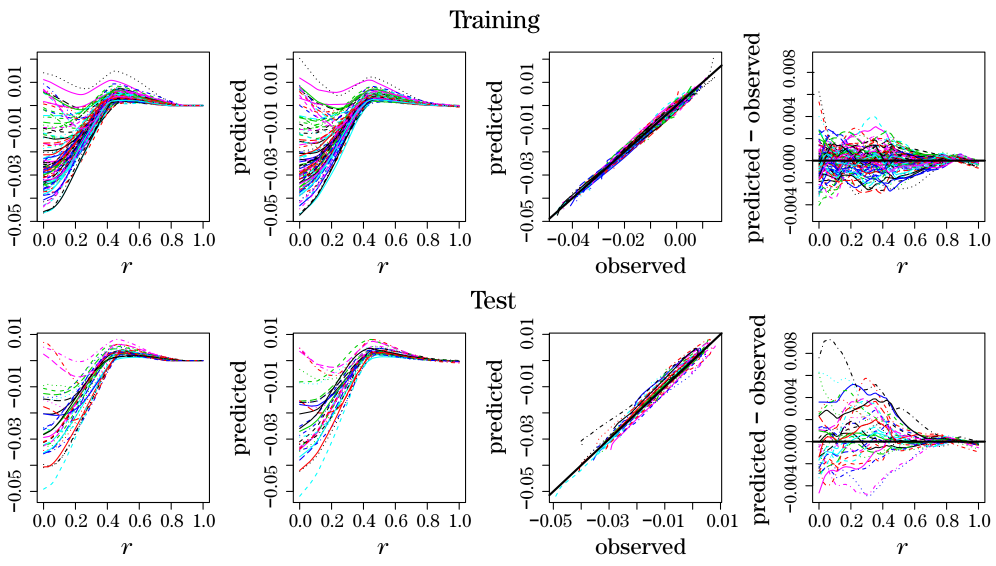

Figure 6. Model fit for the training and test data. From left to right the panels show the model runs (observed data), our prediction (posterior mean) of the model runs, the observed curves against the predicted curves, and residual curves. The top panels show predictions of the training data while the bottom panels show predictions of the test data.

because moving from one posterior mode to another may require a substantial increase in σ^2, which is not likely to be favored in a typical RJMCMC chain. This is where tempering is extremely useful. We chose a temperature ladder through experimentation, and found that the maximum inverse temperature at which we were able to mix well over the model space was 0.001. We found mixing reasonable when we used 100 inverse temperatures in the pattern $t_i = 1 - \Phi\left(i - 50.5/13.2\right)$, where Φ denotes the standard Normal cdf. Plots of predicted curves (point-wise posterior means) and residual curves for the training and test data are given in Figure 6. These plots show that we are able to explain most of the variation in the computer model runs with the emulator. The posterior predictions for four individual curves from the test set are shown in Figure 7 with 95% central regions (curve-wise), constructed following Sun and Genton (2011).

Upon obtaining a suitable BMARS surrogate model, we used the analytical Sobol' decomposition as described above to obtain the sensitivity indices for each posterior sample. First, consider the model sensitivity when we treated the functional variable as one of the inputs. Figure 8 shows the posterior distributions of the sensitivity indices for the most important main effects and interactions

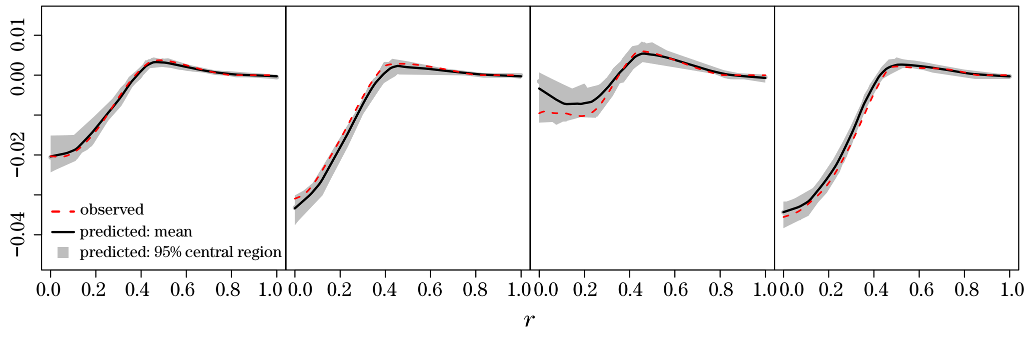

Figure 7. Predicted mean curves with 95% central regions for four model runs from the test data.

as boxplots, as well as the posterior distributions of the total sensitivity indices. Importance was determined by ranking means. These showed that radial position (the functional variable) explains most of the variance, and that the most important input variable is the spacer thickness. These variables have a strong interaction with each other and show up in interactions with other variables. The reality of this problem is that all the effects should be interactions with the functional variable, since there is nearly zero variance in the model output at the large radial positions. However, because of the sequential nature of the Sobol' decomposition (sensitivity indices for interactions are the additional variance explained when the main effects and lower order interactions are already included), it is not surprising to see important terms that do not interact with radial position. It could be that the higher order interaction with radial position only contributes a small amount.

Now, consider the approach to sensitivity analysis that takes sensitivity as a function of radial position. For each main effect and interaction we now have a sensitivity index that is a function of r. Plots of posterior mean sensitivity for the main effects and most important effects are shown in functional pie charts in Figure 9. Also included in Figure 9 are plots of how the standard deviation as a function of radial position is partitioned. Importance was determined by integrating the partitioned variance functions over radial position. We note that the plots are blank when r approaches 0.9 because many of the MCMC draws yield zero variance after that point, and we do not try to decompose zero variance. This is a desirable property, since the actual data have zero variance for these values of r. These plots show us which variables and interactions are most important at different radial positions. Clearly, the spacer thickness is most important

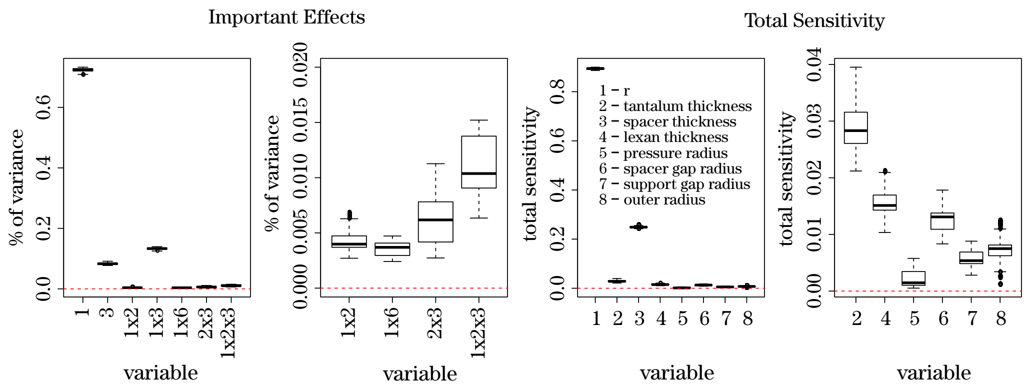

Figure 8. Sensitivity analysis including the functional variable. The first boxplots are the posterior distributions of the largest sensitivity indices, indicating which main effects and interactions explain the most variance. The second plot is an enlarged version to show the sensitivity indices for terms that are hard to compare in the first plot. The next two plots similarly show the posterior distributions of the total indices. Note that the functional variable, labeled as 1, explains most of the variance.

because of the large main effect and the important interactions. These plots also show us that the spacer gap radius plays an important role for $0.3 < r < 0.4$. The lexan thickness plays an important role on its own and interacting with spacer thickness for $r > 0.3$. The tantalum thickness is important in combination with spacer thickness and on its own throughout the range of r. The spacer gap radius plays a small role for $r < 0.2$, the most varied part of the functions.

We see that the pressure radius, support gap radius, and outer radius of the specified ranges do not play an important role in the simulator. Under the specified ranges, it is clear that the spacer thickness deserves the most attention when it comes to designing a protective plate configuration that will be most reliable.

6. Discussion

We have outlined the potential benefits of performing sensitivity analysis using BMARS as an emulator for functional data. We introduced a way of doing sensitivity analysis with functional data. We gave an altered version of BMARS with priors that improve the fit in many circumstances, specifically in cases where the traditional priors lead to overfitting. We also presented modifications for the efficient handling of one dimensional functional data on a fixed grid. We showed that BMARS is fast, accurate, flexible, and can handle datasets

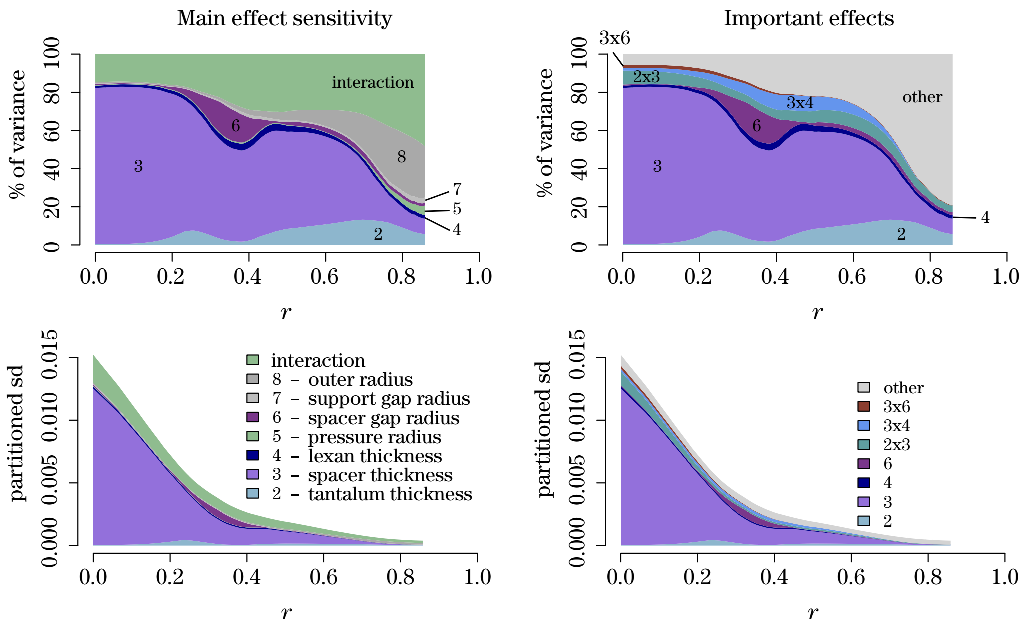

Figure 9. Sensitivity analysis as a function of radial position. The top plots show functional pie charts of the sensitivity indices for the main effects and the most important effects. The bottom plots show the way the actual standard deviation of the model (a function of r) is partitioned. The online version of this figure is in color.

with many variables and thousands of curves, each represented with hundreds of observations. We described how global sensitivity analysis can be performed effortlessly and without Monte Carlo error. We introduced a tempering scheme that leads to more satisfactory posterior sampling than previous approaches. Finally, we stress the fact that the Bayesian nature of the method allows for a full assessment and propagation of the uncertainties.

Automatic specification of a model of this form is an ongoing problem. For instance, using the automatic settings of the BMARS code that accompanies Denison et al. (2002) resulted in extreme overfitting in the plate deformation problem. Our approach requires some tuning in specifying the hyperprior for the mean number of basis functions, and we are interested in other ways of regularizing. Another area for improvement in this model is the assumed homoscedasticity. As the rightmost plots in Figure 6 show, the unexplained variance changes with radial position, though we assumed it is constant. Conceptually, there would be no problem introducing heteroscedasticity by assuming variance depends on the functional variable and learning the form of the variance functions. However, computationally, this is troublesome because we would need to manipulate very large matrices.

Owing to the many computer models coming into use that output large and complex data, we feel that the BMARS methods outlined have promising possibilities in the field of uncertainty quantification.

Supplementary Materials

The derivation of Sobol' indices for the altered Friedman function used in Section 3.4 can be found in the Supplementary Material.

Acknowledgment

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. LLNL-JRNL-716518.

References

Altekar, G., Dwarkadas, S., Huelsenbeck, J. P. and Ronquist, F. (2004). Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics* **20**, 407–415.

Atchadé, Y. F., Roberts, G. O. and Rosenthal, J. S. (2011). Towards optimal scaling of metropolis-coupled markov chain monte carlo. *Statistics and Computing* **21**, 555–568.

Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. A. (1984). *Classification and Regression Trees.* CRC press.

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. and McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association* **93**, 935–948.

Denison, D. G., Holmes, C. C., Mallick, B. K. and Smith, A. F. (2002). *Bayesian Methods for Nonlinear Classification and Regression*, Volume 386. John Wiley & Sons.

Denison, D. G., Mallick, B. K. and Smith, A. F. (1998a). A Bayesian CART algorithm. *Biometrika* **85**, 363–377.

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

Friedamn, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics* **19**, 1–141.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *In Computing Science* *and Statistics: Proceedings of the 23rd Symposium on the Interface (E. M. Keramides, ed.)* 156–163. Interface Foundation, Fairfax Station, Va.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association* **90**, 909–920.

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

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

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

Kolda, T. G. (2006). *Multilinear Operators For Higher-Order Decompositions.* United States. Department of Energy.

Lamboni, M., Makowski, D., Lehuger, S., Gabrielle, B. and Monod, H. (2009). Multivariate global sensitivity analysis for dynamic crop models. *Field Crops Research* **113**, 312–320.

Lev-Ari, H. et al. (2005). Efficient solution of linear matrix equations with application to multistatic antenna array processing. *Communications in Information & Systems* **5**, 123–130.

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**, 410–423.

Maljovec, D., Wang, B., Kupresanin, A., Johannesson, G., Pascucci, V. and Bremer, P.-T. (2013). Adaptive sampling with topological scores. *International Journal for Uncertainty Quantification* **3**.

Neal, R. M. (1996). Sampling from multimodal distributions using tempered transitions. *Statistics and Computing* **6**, 353–366.

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.

Pannell, D. J. (1997). Sensitivity analysis of normative economic models: theoretical framework and practical strategies. *Agricultural Economics* **16**, 139–152.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science* **4**, 409–423.

Saltelli, A. and Annoni, P. (2010). How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software* **25**, 1508–1517.

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.

Saltelli, A., Tarantola, S. and Campolongo, F. (2000). Sensitivity analysis as an ingredient of modeling. *Statistical Science* **15**(4), 377–395.

Saltelli, A., Tarantola, S., Campolongo, F. and Ratto, M. (2004). *Sensitivity Analysis in Practice: a Guide to Assessing Scientific Models.* John Wiley & Sons.

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

Sobol', I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation* **55**, 271–280.

Storlie, C. B., Swiler, L. P., Helton, J. C. and Sallaberry, C. J. (2009). Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models. *Reliability Engineering & System Safety* **94**, 1735–1763.

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.

Sudret, B. (2008). Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety* **93**, 964–979.

Sun, Y. and Genton, M. G. (2011). Functional boxplots. *Journal of Computational and Graphical Statistics* **20**, 316–334.

Surjanovic, S. and Bingham, D. (2015). Virtual library of simulation experiments: Test functions and datasets. Retrieved December 28, 2015, from `http://www.sfu.ca/~ssurjano`.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J. and Morris, M. D. (1992). Screening, predicting, and computer experiments. *Technometrics* **34**, 15–25.

Zellner, A. (1986). On assessing prior distributions and bayesian regression analysis with g-prior distributions. *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti* **6**, 233–243.

Applied Mathematics and Statistics Department, UC Santa Cruz, 1156 High Street, Santa Cruz, CA 95064 USA.

E-mail: dfrancom@ucsc.edu

Applied Mathematics and Statistics Department, UC Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, USA.

E-mail: bruno@soe.ucsc.edu.

Applied Statistics Group, Computational Engineering Division, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA.

E-mail: kupresanin1@llnl.gov

Applied Statistics Group, Computational Engineering Division, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA.

E-mail: gardarj@me.com

(Received March 2016; accepted December 2016)