Bayesian Cubic Spline in Computer Experiments

Yijie Dylan Wang[1] and C. F. Jeff Wu[2]

[1] *Blizzard Entertainment*

[2] *Georgia Institute of Technology*

Abstract: Cubic splines are commonly used in numerical analysis. It has also become popular in the analysis of computer experiments, thanks to its adoption by the software JMP 8.0.2 2010. In this paper a Bayesian version of the cubic spline method is proposed, in which the random function that represents prior uncertainty about y is taken to be a specific stationary Gaussian process and y is the output of the computer experiment. An MCMC procedure is developed for updating the prior given the observed y values. Simulation examples and a real data application are given to show that the proposed Bayesian method performs better than the frequentist cubic spline method and the standard method based on the Gaussian correlation function.

Key words and phrases: Gaussian Process, MCMC, kriging, nugget, uncertainty quantification

1. Introduction

Because of the advances in complex mathematical models and fast computation, computer experiments have become popular in engineering and scientific investigations. Computer simulations can be much faster or less costly than running physical experiments. Furthermore, physical experiments can be hard to conduct or even infeasible when only rare events, like land slides or hurricanes, are observed. There are many successful applications of computer experiments as reported in the literature. The Gaussian process (GP) has been used as the main tool for modelling computer experiments. See the books by Santner, Williams and Notz (2003), Fang, Li and Sudjianto (2005), as well as the November 2009 issue of Technometrics, which was devoted to computer experiments.

First we introduce the GP model. Suppose an experiment involves k factors $\mathbf{x} = (x_1, \ldots, x_k)^t$ and n computer runs are performed at $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. We can write the input as the $n \times k$ matrix $D = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^t$. The corresponding

response values is the vector $\mathbf{Y}_D = (y_1, \ldots, y_n)^t$. The GP model assumes that

$$y(\mathbf{x}) = \mathbf{b}^t \mathbf{f}(\mathbf{x}) + Z(\mathbf{x}), \tag{1.1}$$

where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_s(\mathbf{x}))^t$ is a vector of s known regression functions, $\mathbf{b} = (b_1, \ldots, b_s)^t$ is a vector of unknown coefficients, and $Z(\mathbf{x})$ is a stationary GP with mean zero, variance σ^2 and isotropic correlation function $\text{corr}(y(\mathbf{x}_1), y(\mathbf{x}_2)) = R(\mathbf{x}_1, \mathbf{x}_2) = R(\|\mathbf{x}_1 - \mathbf{x}_2\|)$. For the GP model in (1.1), the best linear unbiased predictor (BLUP) of $y(\mathbf{x})$ is an interpolator, which will be shown in (2.2).

One popular choice of the correlation function is the Gaussian correlation function, which is the product exponential correlation function with power two. For the one-dimension case, it can be written as:

$$R(d) = \exp(-\theta d^2), \tag{1.2}$$

where $d = \|\mathbf{x}_1 - \mathbf{x}_2\|$ is the distance between two input values $\mathbf{x}_1$ and $\mathbf{x}_2$, and θ is the scale parameter. It has been used in many applications (O'Hagan 1978, Sacks and Schiller 1988, Sacks, Schiller, Welch 1989 and Abrahamsen 1997) and software including JMP 8.0.2 2010. However, a process $y(\mathbf{x})$ with (1.2) as the correlation function has the property that its realization on an arbitrarily small, continuous interval determines the realization on the whole real line. This global influence of local data is considered unrealistic and possibly misleading in some applications (Diggle and Ribeiro, 2007, p. 54). We shall refer to this property as *global prediction*. Another well known correlation function is the Matérn family (Matérn, 1960). For the one-dimension case, it is a two-parameter family:

$$R(d) = \{2^{\nu-1}\Gamma(\nu)\}^{-1}(d/\phi)^{\nu} K_{\nu}(d/\phi),$$

where $K_{\nu}(\cdot)$ denotes a modified Bessel function of order $\nu > 0$, and $\phi > 0$ is a scale parameter for the distance d. As $\nu \to \infty$, the Matérn correlation function converges to (1.2).

Another commonly used interpolation method is the spline. An order-s spline with knots ξ_i, $i = 1, ..., l$ is a piecewise-polynomial of order s and has continuous derivatives up to order $s - 2$. A cubic spline has $s = 4$. The GP may also be viewed as a spline in a reproducing kernel Hilbert space, with the reproducing kernel given by its covariance function (Wahba 1990). The main

difference between them is in the interpretation. While the spline is driven by a minimum norm interpolation based on a Hilbert space structure, the GP is driven by minimizing the expected squared prediction error based on a stochastic model.

In this paper, we focus on the cubic spline by considering it in the GP framework via the cubic spline correlation function (Currin et al. 1988, Santner et al. 2003):

$$R(d) = \begin{cases} 1 - 6(\frac{d}{\theta})^2 + 6(\frac{|d|}{\theta})^3, & \text{if } |d| < \frac{\theta}{2}, \\ 2(1 - \frac{|d|}{\theta})^3, & \text{if } \frac{\theta}{2} \le |d| < \theta, \\ 0, & \text{if } |d| \ge \theta, \end{cases} \tag{1.3}$$

where $\theta > 0$ is the scale parameter. Currin et al. (1991) showed that the BLUP with the function in (1.3) as the correlation function gives the usual cubic spline interpolator. An advantage of the cubic spline correlation is that θ can be made small, which permits prediction to be based on data in a local region around the predicting location (Santner et al. 2003, p. 38). We shall refer to this property as *local prediction*.

In this paper, we introduce a Bayesian version of the Gaussian process approach for the cubic spline correlation function given in (1.3). One advantage of Bayesian prediction is that the variability of $y(\mathbf{x})$ given observations can be used to provide measures of posterior uncertainty, and designs can be sought to minimize the expected uncertainty (Ylvisaker 1987, Sacks, Welch, Mitchell, and Wynn 1989). Some empirical studies have shown the superiority of Gaussian processes over the other interpolating techniques, including splines (see Laslett 1994). Here we show some potential advantage of using Bayesian cubic spline in the GP model compared to the power exponential correlation function (1.2) through simulation studies.

The paper is organized as follows. In Section 2, we give a brief review of the kriging technique based on GP models. In Section 3.1, we develop a Bayesian version of the cubic spline method, abbreviated as BCS. In Section 3.2, a nugget parameter is added to the GP model underlying the BCS method when numerical and estimation stability is required. A summary procedure for the BCS is given. In Section 3.3, we consider its extension to high dimensions. In Section 4, BCS is compared with two competing procedures in three simulation examples: a GP based on the cubic spline correlation function in (1.3), abbreviated as CS, and

a GP based on the power exponential Gaussian correlation function in (1.2), abbreviated as GC (CS and GC are explained in more detail in Section 2). In Section 5, we compare the performance of BCS and GC on some real data. Some concluding remarks are given in Section 6.

2. A brief review on kriging

The GP model has been used in geostatistics and is known as kriging (Matheron 1963, Cressie 1992, Diggle and Ribeiro 2007). Kriging is used to analyse spatially referenced data which have the following characteristics (Cressie 1992): The observed values y_i are at a discrete set of sampling locations $\mathbf{x}_i$, $i = 1, \ldots, n$, within a spatial region. The observations y_i are statistically related to the values of an underlying continuous spatial phenomenon. Sacks et al. (1989) proposed kriging as a technique for developing meta models from a computer experiment. Computer experiments produce a response for a given set of input variables. Our methodology development only considers deterministic computer experiments, i.e, the code produces identical answers if run twice using the same set of inputs.

Suppose we want to provide the prediction of a function $y(\mathbf{x})$ at an untried location $\mathbf{x}$, given the observed y values at $D = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^t$. For the Gaussian process in (1.1), the best linear unbiased estimator (BLUE) of $\mathbf{b}$ is:

$$\hat{\mathbf{b}} = (\mathbf{f}_D^t \mathbf{R}_D(\hat{\theta})^{-1} \mathbf{f}_D)^{-1} \mathbf{f}_D^t \mathbf{R}_D(\hat{\theta})^{-1} \mathbf{Y}_D, \tag{2.1}$$

where $\mathbf{f}_D = \mathbf{f}(D) = (\mathbf{f}(\mathbf{x}_i))_{\mathbf{x}_i \in D}$, the dependence on θ is now explicitly indicated in the notation, and $\hat{\theta}$ is an estimate of θ. The BLUP of $\mathbf{Y}_0 = y(\mathbf{x}_0)$ at $\mathbf{x}_0 \in \mathbb{R}$ is:

$$\hat{\mathbf{Y}}_0 = \hat{\mathbf{b}}^t \mathbf{f}_0 + \mathbf{R}_0(\hat{\theta})^t \mathbf{R}_D(\hat{\theta})^{-1} (\mathbf{Y}_D - \hat{\mathbf{b}}^t \mathbf{f}_D), \tag{2.2}$$

where $\hat{\mathbf{b}}$ is given in (2.1), $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0)$ and $\mathbf{R}_0 = (R(\mathbf{x}_0 - \mathbf{x}_1), \ldots, R(\mathbf{x}_0 - \mathbf{x}_n))^t$ is the $n \times 1$ vector of correlations between $\mathbf{Y}_D$ and $\mathbf{Y}_0$. If we denote $\mu_D = \mathbf{b}^t \mathbf{f}_D$ and $\mu_0 = \mathbf{b}^t \mathbf{f}_0$, then (2.2) becomes:

$$\hat{\mathbf{Y}}_0 = \hat{\mu}_0 + \mathbf{R}_0(\hat{\theta})^t \mathbf{R}_D(\hat{\theta})^{-1} (\mathbf{Y}_D - \hat{\mu}_D).$$

One way to estimate θ and σ^2 is by maximum likelihood. Maximum likelihood is a commonly used method for estimating parameters in both computer experiments and spatial process models (Wecker and Ansley 1983; Mardia and Marshall 1984; Currin et al. 1988; Sacks, Schiller, and Welch 1989; Sacks, Welch, Mitchell, and

Wynn 1989). For the power exponential correlation function in (1.2), the estimate of σ^2 yields:

$$\hat{\sigma}^2(\theta) = \frac{1}{n}(\mathbf{Y}_D - \mu_D)'\mathbf{R}_D(\theta)^{-1}(\mathbf{Y}_D - \mu_D).$$

Estimation of θ is usually done by a constrained iterative search. We refer to this method as kriging based on the power exponential Gaussian correlation (GC). If we adopt the cubic spline correlation function in (1.3), the correlation parameter θ is both a scale and truncation parameter. In this case, the estimation method of θ is based on the restricted maximum likelihood method (REML). REML (Patterson and Thompson, 1971) was proposed as a method of obtaining less biased estimates of the variance and covariance parameters than the (unrestricted) maximum likelihood method. We refer to this method as kriging based on the cubic spline correlation function (CS).

3. Bayesian cubic spline

3.1 The Prior and Posterior Processes

With $\mathbf{Y}_D \sim \mathcal{N}(\mu_D, \sigma^2\mathbf{R}_D(\theta))$, we now develop the Bayesian framework for the cubic spline method, where $\mathbf{R}$ is the correlation matrix based on the function in (1.3). We first assign the non-informative priors to μ_D and θ, the conjugate prior to σ^2, and further assume that these priors are independent with each other:

$$\begin{aligned} \mu_D &\propto 1, \\ \sigma^2 \mid \alpha, \beta &\sim \text{InverseGamma}(\alpha, \beta), \\ \theta \mid a &\sim \mathcal{U}(0, a), \\ \beta &\propto 1/\beta, \end{aligned}$$

where θ follows the uniform distribution in $(0, a)$, and β^{-1} has a non-informative prior over $(0, \infty)$. Here θ can be viewed as the *knot location* parameter in the spline literature. In the Bayesian spline literature (Dimatteo et al., 2001, Wang 2008), it is a common practice to assign a uniform prior to the knot location parameter. The hyperparameter a is fixed as:

$$a = \max_{\mathbf{x}_i, \mathbf{x}_j \in D} \|\mathbf{x}_i - \mathbf{x}_j\|.$$

The reason for choosing this particular a is because the function in (1.3) is truncated and equals zero for $|d| \geq \theta$. Because we want the GP to have a local

prediction property, we choose a to be the largest distance among the $\mathbf{x}$ values in D. A simulation study (not reported here) shows that a larger value of a does not change overall performance in estimation. Because an unknown shape parameter α will bring unnecessary complication in the computation, we assume α to be fixed and known.

We use the Markov chain Monte Carlo (MCMC) method to perform the Bayesian computation (Christian and Casella 2004; Gill 2008). It samples from probability distributions by constructing a Markov chain that has the desired distribution as its equilibrium distribution. Gibbs sampling can be implemented to obtain the posterior distribution of μ_D and β:

$$\begin{aligned}\mu_D \mid \mathbf{Y}_D, \theta, \alpha, \beta &\propto \int \mathbb{P}(\mathbf{Y}_D \mid \mu_D, \theta, \sigma^2)\mathbb{P}(\mu_D)\mathbb{P}(\sigma^2 \mid \alpha, \beta)\,\mathrm{d}\sigma \\ &\propto \int \mathcal{N}(\mu_D, \mathbf{R}_D(\theta, \sigma^2)) \times \text{InverseGamma}(\alpha, \beta)\,\mathrm{d}\alpha\mathrm{d}\beta, \qquad (3.1)\end{aligned}$$

$$\begin{aligned}\beta \mid \mathbf{Y}_D, \mu_D, \theta, \alpha &\propto \int \mathbb{P}(\mathbf{Y}_D \mid \mu_D, \theta, \sigma^2)\mathbb{P}(\beta)\mathbb{P}(\sigma^2 \mid \alpha, \beta)\,\mathrm{d}\sigma \\ &\propto \int \mathcal{N}(\mu_D, \mathbf{R}_D(\theta, \sigma^2)) \times \beta^{-1} \times \text{InverseGamma}(\alpha, \beta)\,\mathrm{d}\alpha\mathrm{d}\beta.\end{aligned}$$

However, the parameters θ are embedded into the covariance function $\mathbf{R}$ in (1.3) and have no posterior distribution in closed form. Thus we will sample θ using the Metropolis-Hastings (MH) algorithm (Metropolis et al. 1953, Hastings 1970). The MH algorithm works by generating a sequence of sample values in such a way that, as more and more sample values are produced, the distribution of values more closely approximates the desired distribution. Specifically, at each iteration, the algorithm picks a candidate for the next sample value based on the current sample value. Then, with some probability, the candidate is either accepted (in which case the candidate value is used in the next iteration) or rejected (in which case the candidate value is discarded, and the current value is reused in the next iteration).

Specifically, we sample a new value of θ (denoted as θ_{new}) using a normal density with respect to the existing θ (denoted as θ_{old}). The normal densities work as a *jumping* distribution, because they choose a new sample value based on the current sample. In theory, any arbitrary jumping probability density

$Q(\delta_{old} \mid \delta_{new})$ can work, where δ is the parameter of interest. Here we choose a symmetric jumping density $Q(\delta_{old} \mid \delta_{new}) = Q(\delta_{new} \mid \delta_{old})$ for simplicity. The variance term in the normal distribution is the jumping size from the old sample to the new sample of the MH algorithm. Here we choose the variance to be one. As this jumping size gets smaller, the deviation of new parameters from previous ones should get smaller. The jumping distribution is therefore:

$$\log \theta_{new} \sim \mathcal{N}(\log \theta_{old}, 1). \tag{3.2}$$

After getting θ_{new}, the acceptance ratio is defined as:

$$\begin{aligned} r_1 &= \frac{\mathbb{P}(\mathbf{Y}_D|\mu_D,\theta_{new},\sigma^2)\mathbb{P}(\theta_{new}|a)}{\mathbb{P}(\mathbf{Y}_D|\mu_D,\theta_{old},\sigma^2)\mathbb{P}(\theta_{old}|a)} \\ &= \frac{|\mathbf{R}_D(\theta_{new},\sigma^2)|^{-1/2}\exp\big(-\frac{1}{2}(\mathbf{Y}_D-\mu_D)'\mathbf{R}_D(\theta_{new},\sigma^2)^{-1}(\mathbf{Y}_D-\mu_D)\big)\mathbb{1}_{\theta_{new}\in[0,a]}}{|\mathbf{R}_D(\theta_{old},\sigma^2)|^{-1/2}\exp\big(-\frac{1}{2}(\mathbf{Y}_D-\mu_D)'\mathbf{R}_D(\theta_{old},\sigma^2)^{-1}(\mathbf{Y}_D-\mu_D)\big)\mathbb{1}_{\theta_{old}\in[0,a]}}. \end{aligned}$$

We accept θ_{new} with probability r_1 if $r_1 < 1$. If $r_1 \geq 1$, we accept θ_{new}.

3.2 Nugget Parameter

One possible problem with the kriging approach is the potential numerical instability in the computation of the inverse of the correlation matrix in (2.2). This happens when the correlation matrix is nearly singular. Numerical instability is serious because it can lead to large variability and poor performance of the predictor. The simplest and perhaps most appealing way is to add a nugget effect in the GP modeling. In the spatial statistics literature (Cressie, 1992), a nugget effect is introduced to compensate for local discontinuities in an underlying stochastic process. A well-known precursor is the ridge regression in linear regression analysis. Gramacy and Lee (2012) gave justifications for the use of nugget in GP modeling for deterministic computer experiments. Here we consider the option of adding a nugget parameter in GP model by using ridge regression.

Consider the Gaussian model:

$$Y \sim \mathcal{N}(\mu, \mathbf{R}(\sigma^2, \theta) + \tau^2 I),$$

where τ^2 is the nugget parameter. Adding the matrix $\tau^2 I$ to $\mathbf{R}$ makes the covariance matrix nonsingular and helps stabilize the parameter estimate. We can use the MH sampling to estimate τ^2 by letting $\gamma^2 = \tau^2/\sigma^2$ and assigning a uniform $[0, \kappa]$ prior on γ^2, where κ is fixed and known. We can then replace

$\mathbf{R}_D$ and $\mathbf{R}_0$ in Section 3.1 by $\mathbf{V}_D = \mathbf{R}_D + \gamma^2 I$ and $\mathbf{V}_0 = \mathbf{R}_0 + \gamma^2 I$. To use MH sampling for γ^2, we choose the jumping distribution:

$$\log \gamma^2_{new} \sim \mathcal{N}(\log \gamma^2_{old}, 1), \tag{3.3}$$

with the acceptance ratio:

$$\begin{aligned} r_2 &= \frac{\mathbb{P}(\mathbf{Y}_D|\mu_D,\gamma^2_{new},\theta,\sigma^2)\mathbb{P}(\gamma^2_{new}|\kappa)}{\mathbb{P}(\mathbf{Y}_D|\mu_D,\gamma^2_{old},\theta,\sigma^2)\mathbb{P}(\gamma^2_{old}|\kappa)} \\ &= \frac{|\mathbf{V}_D(\gamma^2_{new},\theta,\sigma^2)|^{-1/2}\exp\big(-\frac{1}{2}(\mathbf{Y}_D-\mu_D)'\mathbf{V}_D(\gamma^2_{new},\theta,\sigma^2)^{-1}(\mathbf{Y}_D-\mu_D)\big)\mathbb{1}_{\gamma^2_{new}\in[0,\kappa]}}{|\mathbf{V}_D(\gamma^2_{old},\theta,\sigma^2)|^{-1/2}\exp\big(-\frac{1}{2}(\mathbf{Y}_D-\mu_D)'\mathbf{V}_D(\gamma^2_{old},\theta,\sigma^2)^{-1}(\mathbf{Y}_D-\mu_D)\big)\mathbb{1}_{\gamma^2_{old}\in[0,\kappa]}}. \end{aligned}$$

In our computation, we use the criterion introduced by Peng and Wu (2014) to determine whether or not to include a nugget effect. We use the condition number of a square matrix as the primary measure of singularity, that is, the ratio of its maximum eigenvalue over its minimum eigenvalue. Here we use the LAPACK reciprocal condition estimator in MATLAB to determine whether the covariance matrix $\mathbf{R}$ is ill-conditioned. If $(\kappa_1(\mathbf{R}))^{-1} < 10\epsilon$, where $\epsilon = 2^{-8}$ is the floating-point relative accuracy, then $\mathbf{R}$ is ill-conditioned and we will introduce the nugget effect into the model. Otherwise, we will set the nugget effect to be zero.

With the option of adding a nugget parameter, the steps to perform the Bayesian cubic spline are summarized as follows:

1. Set initial values for μ_D, θ, σ^2 and let $\gamma^2 = 0$.
2. Calculate $\kappa_1(\mathbf{R})$.
3. If $(\kappa_1(\mathbf{R}))^{-1} \geq 10\epsilon$, where ϵ is a specified constant, set $\gamma^2 = 0$, sample μ_D and θ from (3.1), (3.2) respectively. If the parameters do not converge, go back to step 2.
4. If $(\kappa_1(\mathbf{R}))^{-1} < 10\epsilon$, use $\mathbf{V} = \mathbf{R} + \gamma^2 I$ instead of $\mathbf{R}$ and sample μ_D, θ and γ^2 from (3.1), (3.2) and (3.3) respectively. Repeat this step until convergence.
5. Calculate the estimate of Y_0 using (2.2) with μ_D, θ and γ^2.

In step 1, we choose the posterior mode as the starting value. This choice is sensible and can avoid the need of doing "burn-in" in MCMC. See its justification in Geyer (2011).

3.3 Extension to High Dimensions

For multiple dimensions, let $\mathbf{x} \in \mathbb{R}^k$ and assume the correlation function $\mathbf{R}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{t=1}^{k} \mathbf{R}_t(\mathbf{x}_{i,t} - \mathbf{x}_{j,t}) = \prod_{t=1}^{k} \mathbf{R}_t(d_t)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are in $\mathbb{R}^k$, d_t is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ on the tth dimension and $\mathbf{R}_t$ is the correlation function for the tth dimension.

The multi-dimensional spline correlation function $\mathbf{R}(d)$ is the product of the one-dimension spline correlation function with individual parameter θ_t estimated for each dimension (Ylvisaker 1975, Chen, Gu, and Wahba 1989). The corresponding Bayesian computation is done by doing MH sampling for each dimension until convergence. Our criterion for convergence is when the change of $\|\theta\|$ between consecutive iterations of the MCMC computation is smaller than 10^{-4}. In our simulation studies, θ converges within three minutes.

4. Simulation study and results

First, we compare the performance of the proposed Bayesian cubic spline (BCS) method with two other methods: GC and CS (described in Section 2). The criterion for evaluating the performance of the estimators is the integrated mean squared error (IMSE), defined as:

$$\text{IMSE}(\hat{f}) = \int_{\Omega} (\hat{f}(\mathbf{x}) - f(\mathbf{x}))^2 d\mathbf{x},$$

where f and $\hat{f}$ are respectively the true function values and estimated values and Ω is the region of the $\mathbf{x}$ values. The following mean squared error (MSE) is a finite-sample approximation to the IMSE:

$$\text{MSE}(\hat{f}) = \frac{1}{m} \sum_{i=1}^{m} (\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2, \tag{4.1}$$

where m is the number of randomly selected points $\{\mathbf{x}_i\}$ from Ω. Three choices of the true function $f(x)$ are considered in Examples 1-3, which range from low to high dimensions and from smooth to non-smooth functions.

Example 1.

$$f_1(\mathbf{x}) = \{1 - \exp(-.5/x_2)\} \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}.$$

This two-dimensional function is from Currin et al. (1991), where $\mathbf{x} \in [0, 1]^2$ and $f_1 \in [4.1, 13.8]$. We scale f_1 into $[0, 1]$. Currin et al. (1991) studied a 16-run

design in their paper. Four designs are considered here: 4^2 design (16 runs) with levels $(.125, .375, .628, .875)$ (Joseph 2006) and $(0, .3333, .6667, 1)$ (Currin et al. 1991), 5^2 design (25 runs) with levels $(0, .25, .5, .75, 1)$, and 6^2 design (36 runs) with levels $(0, 0.2, 0.4, 0.6, 0.8, 1)$. Four types of noise ϵ are added to $f_1(\mathbf{x})$: $\mathcal{U}(0, 0)$ (no noise), $\mathcal{U}(0, .2)$, $\mathcal{U}(0, .5)$ and $\mathcal{U}(0, 1)$. As the range of the noise increases from 0 to 1, the function $f_1(\mathbf{x}) + \epsilon$ becomes more rugged. It allows us to compare the performance of the three methods as the true function becomes less smooth.

For noise based on $\mathcal{U}(0, 0)$ (and $\mathcal{U}(0, .2)$, $\mathcal{U}(0, .5)$ and $\mathcal{U}(0, 1)$ respectively), we conduct the simulation as follows. First, a noise is randomly sampled from $\mathcal{U}(0, 0)$ (and $\mathcal{U}(0, .2), \mathcal{U}(0, .5)$ and $\mathcal{U}(0, 1)$). For each simulation, the noise is fixed and denoted as $\{\epsilon_1, \ldots, \epsilon_n\}$. Here n, the number of design points, is 16, 16, 25 and 36 respectively for the four designs. Second, the values of $\{f_1(\mathbf{x}_1), \ldots, f_1(\mathbf{x}_n)\}$ are calculated. Then the values of $\{f_1(\mathbf{x}_1)+\epsilon_1, \ldots, f_1(\mathbf{x}_n)+\epsilon_n\}$ are treated as the response values by GC, BCS and CS in parameter estimation. The purpose of this step is to facilitate the study of the robustness of estimation against noise of various sizes. Then, the MSE (see (4.1)) is calculated on $m = 100$ of $\mathbf{x}$ randomly sampled from $[0, 1]^2$. We sample repeatedly and independently $\{\epsilon_1, \ldots, \epsilon_n\}$ and $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ for each simulation and record the average of the MSE values from 500 simulations for each noise and design setting in Table 4.1. For each simulation setting, the method with the smallest average MSE is highlighted in boldface.

The MCMC iterations of the BCS terminate if the change of parameter estimate is smaller than 10^{-4}. The running time takes about 20 seconds for an Intel Xeon CPU with 2.66 GHz and 3.00 GB of RAM to reach convergence. The power exponential method performs best when the noise is small ($\mathcal{U}(0, 0)$ and $\mathcal{U}(0, .2)$) or the design size is large (36-run). This is because the example is relatively smooth when noise is small, and the function f_1 contains the exponential term $\exp(-.5/x_2)$, which is best captured by the non-zero exponential correlation function. GC also benefits from the larger sample size of 6^2, which helps to stabilize the estimate. For relatively small designs (16- and 25-run) with large noise ($\mathcal{U}(0, .5)$ and $\mathcal{U}(0, 1)$), CS and BCS perform better than GC. When the design is small and noise is large, the response surface tends to be very rugged and there is not enough data for GC to estimate the surface with good precision. A localized estimate like CS and BCS with truncated correlation function give

smaller MSE. BCS in most cases outperforms CS. The over-smoothing property of GC can result in large estimation errors as will be shown in Example 2 and in Section 5.

Table 4.1: Average MSE Values for GC, BCS and CS Predictors in Example 1

		$\mathcal{U}(0,0)$	$\mathcal{U}(0,.2)$	$\mathcal{U}(0,.5)$	$\mathcal{U}(0,1)$
	GC	**1.0034**	**1.0437**	1.7192	1.8951
4^2(J)	BCS	1.0901	1.1024	1.6722	**1.8642**
	CS	1.1036	1.2518	**1.6714**	1.9854
	GC	**1.1223**	**1.3495**	1.6545	2.4491
4^2(C)	BCS	1.1651	1.4475	**1.5819**	**2.1270**
	CS	1.1710	1.4322	1.6318	2.1476
	GC	**0.9087**	**1.0186**	1.3516	1.4845
5^2	BCS	1.0112	1.0574	**1.2416**	**1.3391**
	CS	1.0481	1.1204	1.2665	1.5816
	GC	**0.2171**	**0.2588**	**0.5604**	**0.6352**
6^2	BCS	0.2716	0.3079	0.7008	0.6893
	CS	0.2942	0.2769	0.7479	0.8042

Example 2.

$$f_2(x) = 0.3\exp^{-1.4x}|\cos(10\pi x)| + 3x.$$

This is a one-dimension function and contains a non-smooth term $|\cos(10\pi x)|$. Here f_2 is scaled into $[0,1]$. As in Example 1, four types of random noise are added to f_2 and 5, 10, 20 and 30 design points (i.e., $\{x_1,\ldots,x_n\}$ values) are uniformly sampled from $[0,1]$. In each simulation, noise is sampled and fixed, denoted as $\{\epsilon_1,\ldots,\epsilon_n\}$, where $n = 5$ (and 10, 20 and 30 respectively). Then 5 (and 10, 20 and 30) design locations $\{x_1,\ldots,x_n\}$ are uniformly sampled from $[0,1]$. The values of $\{f_2(x_1)+\epsilon_1,\ldots,f_2(x_n)+\epsilon_n\}$ are used as the response values. The MSE for each simulation is calculated on $m = 100$ randomly sampled x values in $[0,1]$. For each design, we repeat this procedure 500 times by taking random samples of $\{\epsilon_i\}_{i=1}^n$ and $\{\mathbf{x}_i\}_{i=1}^n$. The average MSE based on the 500 simulations is given in Table 4.2 for each noise and design setting . Again, the method with the smallest average MSE is highlighted in boldface.

In all cases, CS and BCS beat GC even when no noise is added to the true function. BCS performs better than CS in most cases. CS performs better than

BCS in four cases, three of which the difference is not significant. GC gives much worse results when the design size is small (5, 10) and the noise is large ($\mathcal{U}(0, .5)$ and $\mathcal{U}(0, 1)$). This is due to the global prediction property of GC. For non-smooth functions, this can bring in unnecessarily large errors. On the other hand, the better performance of CS and BCS benefits from their local prediction property.

Table 4.2: Average MSE Values for GC, BCS and CS Predictors in Example 2

		$\mathcal{U}(0,0)$	$\mathcal{U}(0,.2)$	$\mathcal{U}(0,.5)$	$\mathcal{U}(0,1)$
	GC	0.0026	0.2857	0.7214	0.8147
5	BCS	**0.0018**	0.0772	**0.2135**	**0.2740**
	CS	0.0021	**0.0518**	0.2768	0.2853
	GC	0.0017	0.0389	0.1626	0.3260
10	BCS	**0.0013**	0.0261	**0.0508**	**0.1892**
	CS	**0.0013**	**0.0259**	0.0591	0.1964
	GC	0.0015	0.0092	0.1443	0.1547
20	BCS	**0.0004**	**0.0057**	**0.0721**	**0.1208**
	CS	0.0011	0.0063	0.1125	0.1248
	GC	0.0011	0.0049	0.0754	0.1853
30	BCS	**6.70E-04**	0.0032	**0.0341**	**0.1807**
	CS	7.20E-04	**0.0028**	0.0596	0.2005

Example 3.

$$f_3(\mathbf{x}) = \frac{2\pi x_1(x_2 - x_3)}{\ln(x_4/x_5)[1 + \frac{2x_1x_6}{\ln(x_4/x_5)x_5^2x_7} + \frac{x_1}{x_8}]}.$$

This is an 8-dimensional smooth function from Morris et al. (1993), where $x_1 \in [63070, 115600]$, $x_2 \in [990, 1110]$, $x_3 \in [700, 820]$, $x_4 \in [100, 5000]$, $x_5 \in [.05, .15]$, $x_6 \in [1120, 1680]$, $x_7 \in [9855, 12046]$ and $x_8 \in [63.1, 116]$. It is also referred to as the "borehole" data in the literature. Here we scale $x_1, \ldots, x_8$ and f_3 into $[0, 1]$. Morris et al. (1993) proposed a 10-run design with two levels 0 and 1 based on the maximin distance criterion (see Table 4.3). In the study, we consider the 10-run design together with 10-, 20- and 50-run Latin hypercube designs (McKay et al. 1979). A n-run Latin hypercube design in $[0, 1]^k$ is based on the Latin hypercube sampling. For each dimension, we independently sample n values randomly from each interval $(0, 1/n)$, ..., $(1 - 1/n, 1)$ and randomly

permute the n values. Here we apply the maximin criterion to choose the Latin hypercubes, i.e., maximizing the minimum distance between points. As before, four types of noise are added to the true function. In each simulation, after the noise $\{\epsilon_1, \ldots, \epsilon_n\}$ is sampled, one Latin hypercube design is generated, denoted by $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $n = 10, 20, 50$. Then apply GC, CS and BCS to $\{f_1(\mathbf{x}_1) + \epsilon_1, \ldots, f_1(\mathbf{x}_n) + \epsilon_n\}$ for parameter estimation. The MSE is calculated based on $m = 5000$ random samples $\{\mathbf{x}_i\}_{i=1}^{5000}$ in $[0, 1]^8$. The simulations are repeated 1000 times and the average MSE values are given in Table 4.4. The running time for each simulation of BCS is less than 2 minutes on the same machine.

The results are similar to those of Example 1. This is expected as they are both smooth functions. GC gives best results among the three methods when the noise is small ($\mathcal{U}(0, 0)$ and $\mathcal{U}(0, .2)$) or the sample size is large (50LH). BCS and CS perform well when sample size is small (10 and 20) and the noise is large ($\mathcal{U}(0, 1)$). BCS generally outperforms CS. Noting that x_4 and x_5 appear in the f_3 function through the ln function, we have run an alternative simulation by taking the log transformation of x_4 and x_5 and rescale them into $[0, 1]$. Since it does not change the overall picture of comparisons, we do not report these results in the paper.

Table 4.3: A Maximin Distance Design in $[0, 1]^8$ for n=10 (Morris et al. 1993)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1	1	0	0	1	0	1	1
1	1	1	1	0	0	1	0
1	0	0	1	1	0	0	0
0	1	0	0	1	1	0	0
1	1	0	1	0	1	0	1
0	1	1	0	0	0	0	1
0	0	1	1	1	0	1	1
0	0	0	0	0	1	1	1
0	0	1	1	0	1	0	0
1	0	1	0	1	1	1	0

5. Application

Instead of simulating from known functions, we perform another comparison study by using the methane combustion data from Mitchell and Morris (1993).

Table 4.4: MSE Values for GC, BCS and CS Predictors in Example 3

		$\mathcal{U}(0,0)$	$\mathcal{U}(0,.2)$	$\mathcal{U}(0,.5)$	$\mathcal{U}(0,1$
	GC	**0.0386**	**0.0457**	**0.0695**	0.1850
10	BCS	0.0490	0.0516	0.0855	**0.1608**
	CS	0.0501	0.0672	0.0884	0.1691
	GC	**0.0277**	**0.0473**	**0.0721**	0.1821
10LH	BCS	0.0473	0.0655	0.0890	0.1542
	CS	0.0458	0.0632	0.0876	**0.1409**
	GC	**0.0013**	**0.0125**	**0.0405**	0.1714
20LH	BCS	0.0067	0.0366	0.0784	**0.1509**
	CS	0.0077	0.0298	0.0868	0.1621
	GC	**5.10E-04**	**0.0096**	**0.0311**	**0.1355**
50LH	BCS	0.0039	0.0117	0.0520	0.1428
	CS	0.0043	0.0159	0.0581	0.1523

Table 5.5 shows its 50-run design. In addition, Mitchell and Morris gave 20-, 30-, 40- and 50-run, 7-variable maximin designs in their paper. The first 20, 30, 40 and 50 runs in Table 5.5 consist of these designs, which we denote as D20, D30, D40 and D50. The response y is the logarithm of the ignition delay time.

Before conducting the comparison, a careful data analysis is performed to show some feature of the data. For D20, D30 and D50, we randomly take 90%, 80% and 50% of the original data as the response values for GC and BCS to estimate θ. The average values of $\hat{\theta}_j$, $j = 1, \ldots, 7$, based on 100 simulations, are calculated and given in Table 5.6 for each setting. For each design, the value of $\hat{\theta}$ from GC increases as the number of input data decreases, while the $\hat{\theta}$ values for BCS are more stable. The divergent behaviour between GC and BCS for this data can be partially explained by their respective global and local prediction properties. First, note that a larger θ value indicates a smoother surface. As the size of input data gets smaller, the data points are spread more thinly in the design region $[0, 1]^7$. The fitted response surface by GC will become more smooth due to its global prediction property. The change will not be as dramatic for BCS thanks to its local prediction property. Even though we do not know what the true response surface is or how rugged it is, this divergent behavior seems to suggest that BCS is a better method for the data. This is confirmed in

the next study based on cross validation.

Table 5.5: Methane Combustion Data

Run	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
1	0	0	0	0.5	1	1	0.25	7.9315
2	0.25	0.5	0.5	0.75	0	1	0	6.2171
3	0	1	0	0.25	0	0	1	7.8535
4	0.5	0.5	0.75	0	1	0.25	0	7.5708
5	0	0.75	0.75	1	1	1	0.5	6.3491
6	1	0	1	0.25	0	1	0	5.3045
7	0	1	0	0.75	1	0.5	1	8.5372
8	0.75	0.25	0	1	1	0	1	7.871
9	0.5	0.75	0.25	0	0.25	0.5	0.5	7.8725
10	0.25	1	0.75	0.75	0.5	0	0.25	6.593
11	0.5	0	1	0.25	1	0.75	1	6.2131
12	1	0	0	0.5	0.5	0.5	1	7.6311
13	1	0.5	1	0.75	0	0.25	0.5	5.109
14	0	1	0.25	0.25	0.75	1	0	8.4206
15	1	1	0	1	0.25	0	0.5	7.2242
16	0.5	0	0.25	1	0	0.25	0.75	6.0216
17	1	1	1	1	0.5	1	0	5.3495
18	1	1	0.5	0.25	0	1	1	6.0325
19	1	0	1	0	0.75	0	0.25	6.4065
20	0.5	1	0.75	1	0.25	0.75	1	5.5674
21	0.25	0	0.5	0.25	0.25	0	1	6.5214
22	0	0.5	0.5	0	0.75	0.5	0.75	7.7907
23	0.25	0	0	0.25	0	0.75	0.5	7.3542
24	0.75	0.75	1	0	0	0	0	5.8651
25	0.25	0	1	0.5	0.75	0.5	0	6.4489
26	0.75	1	0.25	0.75	1	0.75	0.25	7.6225
27	0	1	1	0.5	0.25	1	0.5	5.8572
28	1	0.5	1	0	1	1	0.5	6.5656

Table 5.5 – *Continued from previous page*

Run	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
29	1	0.25	1	1	1	0.25	0	5.7137
30	0	0	0	1	0.25	1	1	6.5603
31	0.5	0	0.5	0	0.75	1	0.25	7.5044
32	1	0	0.75	0.75	0.5	0.75	0.25	5.8721
33	1	0	0	0	1	0.25	0	8.206
34	1	0.5	0	1	0	0.75	1	6.3746
35	0.25	0.5	1	0.75	0.5	1	1	5.4478
36	1	0.75	0	0	0.5	1	0.75	7.6953
37	0.5	0	1	0	0	0.5	0.75	5.3423
38	0.5	1	0	1	0	1	0.25	6.4493
39	1	0.25	0	0.5	0	0.5	0	6.8957
40	0.75	0.5	0.25	0.5	1	1	1	7.5563
41	0.75	0.75	0.25	0.5	0	0.25	1	6.7549
42	0.75	0	0.75	0.75	0	1	1	5.0056
43	0	0.25	0.25	1	0.75	0.25	0.75	7.4006
44	1	0.25	0.75	0	0.25	0	1	5.6656
45	0.5	0.5	0.5	0.5	0.5	0	0	7.4111
46	0.75	1	0.75	0.25	0.25	0.75	0.25	6.7111
47	1	0.5	0.25	0.25	0.75	0.75	0	7.9182
48	1	0.25	0.5	1	1	1	0.5	6.2543
49	0	0.75	0.5	0.75	0.25	0.25	0.5	6.7319
50	0.25	1	0.25	1	0.75	1	0.75	6.9749

We now use the same data and design settings to run cross validations on D50, D40 and D30 for each of the three methods. One round of cross-validation involves partitioning the data into complementary subsets, performing the analysis on one subset (called the *training* set), and validating the analysis on the other subset (called the *validation* set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation

Table 5.6: Average $\hat{\theta}$ values using GC and BCS

	Method	% of input	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	$\hat{\theta}_6$	$\hat{\theta}_7$
D20	GC	90	0.1137	0.0368	0.2697	0.0606	0.0712	0.0101	0.0137
D20	GC	80	0.1892	0.0878	0.4891	0.0928	0.1082	0.0348	0.0647
D20	GC	50	0.4092	0.2977	0.7952	0.3177	0.3580	0.2476	0.2681
D20	BCS	90	4.1345	4.0635	3.6870	3.8788	3.8440	4.2943	3.3749
D20	BCS	80	4.5264	4.5190	3.6424	3.8407	4.5733	4.6059	4.0459
D20	BCS	50	4.1244	4.8617	3.8581	3.4646	4.2030	5.5327	3.1355
D30	GC	90	0.0958	0.0318	0.1582	0.0335	0.0700	0.0079	0.0144
D30	GC	80	0.1216	0.0647	0.3231	0.0667	0.1017	0.0292	0.0256
D30	GC	50	0.2293	0.1874	0.5877	0.1790	0.2252	0.1315	0.1348
D30	BCS	90	3.7533	4.9328	4.6481	3.6454	5.2628	4.6647	2.0442
D30	BCS	80	3.4020	5.2822	5.4022	3.4501	5.8690	4.6797	2.5237
D30	BCS	50	4.5891	4.9748	5.8604	3.4380	5.5703	4.9078	2.4123
D50	GC	90	0.1297	0.0441	0.1057	0.0394	0.0529	0.0112	0.0129
D50	GC	80	0.1189	0.0404	0.1676	0.0506	0.0741	0.0093	0.0124
D50	GC	50	0.1200	0.0449	0.2086	0.0641	0.0956	0.0123	0.0155
D50	BCS	90	3.3356	4.9131	4.9231	3.2469	3.8125	3.5908	5.0959
D50	BCS	80	3.3427	5.4804	4.3651	3.4601	3.6676	3.3073	4.9948
D50	BCS	50	3.8516	4.7773	4.5888	3.6960	3.8341	3.3936	5.3051

results are averaged over the rounds (Geisser 1993 and Kohavi 1995). Each time we take a fixed number of data out of D50 (and D40, D30 respectively) and use them for model fitting. The remaining data are used to calculate the MSE in (4.1). Because CS gives much larger MSE in each case, we only report the MSE results for GC and BCS. For D50, the results of training data size of 40 and 30 are plotted in Figures 5.1 and 5.2. In each figure, one dot indicates the MSE from GC versus the MSE from BCS for a given design. The reference line of 45° indicates that the two designs are equally good since they render the same MSE. When the majority of the dots is below the line, it means BCS has smaller MSE. This is evident in Figure 5.2. GC gives some very bad predictions with MSE as high as 5.5 (dots in right bottom of Figure 5.2), while the majority of MSE of BCS center around 1.5. The average of MSE from 100 simulations for each cross validation setting for BCS and GC and the percentage of BCS outperforming GC are given in Table 5.7. There is a much larger difference between GC and BCS when the training data size is relatively small (20 for D30, 25 for D40 and 30 for D50). This is probably caused by the global prediction and over-smoothing properties of GC.

Table 5.7: Comparison of GC and BCS

Design	Training Data Size	MSE(GC)	MSE(BCS)	% Improvement
D50	40	0.6237	0.6524	58
D50	30	2.4980	1.3908	92
D40	30	0.8812	0.7132	61
D40	25	2.8516	1.5590	89
D30	25	1.6108	0.7888	69
D30	20	1.9498	1.2729	86

6. Conclusions

The cubic spline is widely used in numerical approximation. In GP modeling, use of the cubic spline correlation function in (1.3) can lead to a sparse correlation matrix with many zero off-diagonal elements. By comparison, the two commonly used correlation functions, the Matérn family and the power exponential correlation, do not enjoy this property. A sparse correlation matrix can reduce the cost of computation and enhance its stability. The viability of

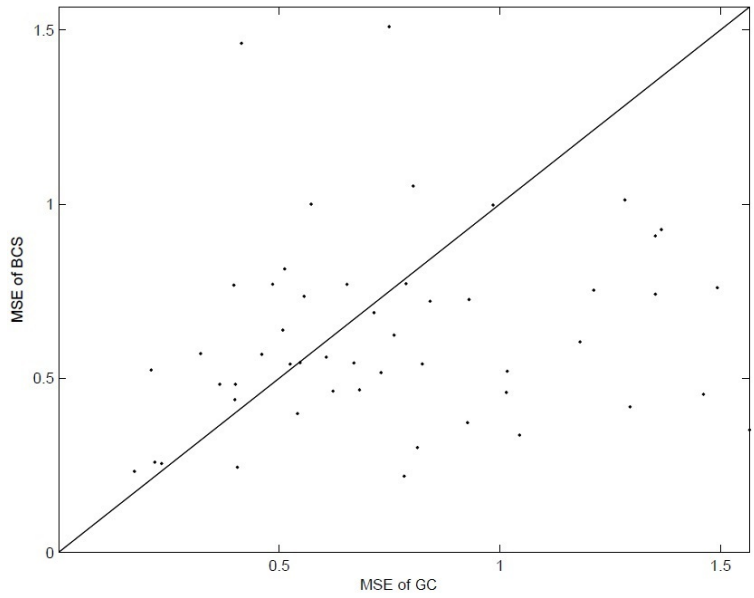

Figure 5.1: MSE of 40 Training Data in D50

cubic spline for computer experiment applications received a further boost when JMP 8.0.2 2010 (and its update 11.1.1 2014) provided the power exponential correlation and the cubic spline correlation as its *only* two choices in GP modeling. The prominence that JMP gives to the cubic spline was one motivation for us to develop a Bayesian version of the cubic spline method. By putting a prior on the parameters in the cubic spline correlation function in (1.3), Bayesian computation can be performed by using MCMC. The Bayesian cubic spline should outperform its frequentist counterpart because of its smoothness and shrinkage properties. It also provides posterior estimates, which enable statistical inference on the parameters of interest.

In this paper we compare BCS with CS and GC in three simulation examples and application to real data. We have also considered other correlation functions with compact support such as the spherical family:

$$R(d) = \begin{cases} 1 - \frac{3|d|}{2\theta} + \frac{1}{2}(\frac{|d|}{\theta})^3, & \text{if } |d| \le \theta, \\ 0, & \text{if } |d| > \theta. \end{cases}$$

Because the performance of the frequentist version of the spherical family is

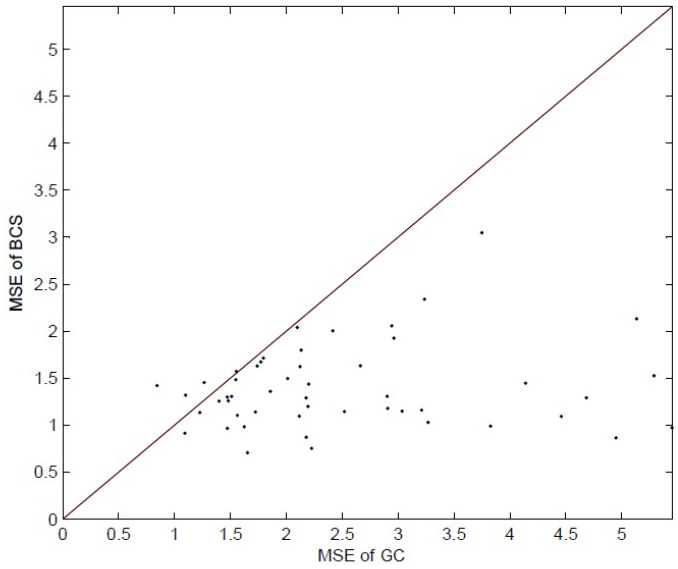

Figure 5.2: MSE of 30 Training Data in D50

similar to that of the cubic spline, these results are omitted in the paper. In the three simulation examples, BCS outperforms CS in most cases. GC performs the best when the true function is smooth or the data size is large. BCS and CS perform better than GC when the true function is rugged and the data size is relatively small. This difference in performance can be explained by the local prediction property of BCS and CS and the global prediction property of GC. Recall that, in global prediction, the prediction at any location is influenced by far-away locations (though with less weights). This leads to over-smoothing, which is not good for a rugged surface. Local prediction does not suffer from this as the prediction depends only on nearby locations. In the real data application, BCS outperforms GC in all choices of design. In summary, when the response surface is non-smooth and/or the input dimension is high, the BCS method can have potential advantages and should be considered for adoption in data analysis.

There are other methods that also attempt to balance between local and global prediction. See, for examples, Kaufman et al. (2011) and Gramacy and Apley (2014), although the main purpose of these two papers was to leverage

sparsity to build emulators for large computer experiments. Comparisons of our proposed method with these and other methods in the literature will require a separate study and it lies outside our original scope of comparing the Bayesian and the frequentist methods for cubic spline.

Some issues need to be considered in future work. When the dimension is high, the parameter estimation is based on the MH sampling which can be costly. Grouping parameters to reduce computation is an alternative. Also, we have considered mostly non-informative priors. If more information is available, informative prior assignments should be considered.

Acknowledgments

The paper was part of Y. Wang's doctoral thesis at Georgia Tech under the supervision of Jeff Wu. The co-author Wu would like to dedicate this paper to the celebration of Professor George Tiao's 80th birthday and to record his appreciation and deep gratitude to George for his tutelage and friendship over more than 35 years. The authors would like to thank Roshan Joseph and Simon Mak for their valuable comments. The research was supported by NSF grant DMS 1308424 and DOE grant DE-SC0010548.

Bibliography

[1] Petter Abrahamsen. *A Review of Gaussian Random Fields and Correlation Functions*. Norsk Regnesentral/Norwegian Computing Center, 1997.

[2] George Casella and Edward I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[3] Zehua Chen, Chong Gu, and Grace Wahba. Comment on "linear smoothers and additive models". *The Annals of Statistics*, 17(3):515–521, 1989.

[4] Noel Cressie. *Statistics for Spatial Data*. Wiley, 1992.

[5] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. A Bayesian approach to the design and analysis of computer experiments. *ORNL-6498*, 1988.

[6] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

[7] Peter J. Diggle and Paulo J. Ribeiro. *Model-based Geostatistics*. Springer, 2007.

[8] Ilaria DiMatteo, Christopher R. Genovese, and Robert E. Kass. Bayesian curve-fitting with free-knot splines. *Biometrika*, 88(4):1055–1071, 2001.

[9] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and Modeling for Computer Experiments*. Chapman and Hall/CRC, 2010.

[10] Seymour Geisser. *Predictive Inference: an Introduction*, volume 55. CRC Press, 1993.

[11] Alan E. Gelfand and Adrian F.M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[12] Charles J. Geyer. *Introduction to Markov Chain Monte Carlo.* Handbook of Markov Chain Monte Carlo, CRC Press, 2011.

[13] Jeff Gill. *Bayesian Methods: A Social and Behavioral Sciences Approach.* CRC press, 2002.

[14] Robert B. Gramacy and Daniel W. Apley. Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 2014 (to appear in print).

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

[16] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[17] V. Roshan Joseph. Limit kriging. *Technometrics*, 48(4):458–466, 2006.

[18] Cari G. Kaufman, Derek Bingham, Salman Habib, Katrin Heitmann, and Joshua A. Frieman. Efficient emulators of computer experiments Using compactly supported correlation functions, with an application to cosmology. *Annals of Applied Statistics*, 5(4):2470–2492, 2011.

[19] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145. Lawrence Erlbaum Associates Ltd, 1995.

[20] Geoffrey M. Laslett. Kriging and splines: an empirical comparison of their predictive performance in some applications. *Journal of the American Statistical Association*, 89(426):391–400, 1994.

[21] Kanti V. Mardia and R.J. Marshall. Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*, 71(1):135–146, 1984.

[22] Bertil Matérn. Spatial variation. stochastic models and their application to some problems in forest surveys and other sampling investigations. *Meddelanden fran statens Skogsforskningsinstitut*, 49(5), 1960.

[23] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

[24] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087, 1953.

[25] Max D. Morris, Toby J. Mitchell, and Donald Ylvisaker. Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[26] Anthony O'Hagan and J.F.C. Kingman. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological)*, 2:1–42, 1978.

[27] H. Desmond Patterson and Robin Thompson. Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58(3):545–554, 1971.

[28] Chien-Yu Peng and C.F. Jeff Wu. On the choice of nugget in kriging modeling for deterministic computer experiments. *Journal of Computational and Graphical Statistics*, 23(1):151-168, 2014.

[29] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*, volume 319. Citeseer, 2004.

[30] Jerome Sacks and Susannah Schiller. Spatial designs. *Statistical Decision Theory and Related Topics IV*, 2(32):385–399, 1988.

[31] Jerome Sacks, Susannah Schiller, and William J. Welch. Designs for computer experiments. *Technometrics*, 31(1):41–47, 1989.

[32] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[33] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer Verlag, 2003.

[34] Grace Wahba. *Spline Models for Observational Data*, volume 59. Society for Industrial and Applied Mathematics, 1990.

[35] Xiao Wang. Bayesian free-knot monotone cubic spline regression. *Journal of Computational and Graphical Statistics*, 17(2):518–527, 2008.

[36] William E. Wecker and Craig F. Ansley. The signal extraction approach to nonlinear regression and spline smoothing. *Journal of the American Statistical Association*, 78(381):81–89, 1983.

[37] Donald Ylvisaker. Designs on random fields. *A Survey of Statistical Design and Linear Models*, 37(6):593–607, 1975.

[38] Donald Ylvisaker. Prediction and design. *The Annals of Statistics*, 52(17):1–19, 1987.

Yijie Dylan Wang
Blizzard Entertainment
E-mail: (dylan.jie@gmail.com)
C. F. Jeff Wu
Georgia Institute of Technology
E-mail: (jeffwu@isye.gatech.edu)