

GPFIT: A NEW R PACKAGE FOR FITTING GAUSSIAN PROCESS MODELS TO DETERMINISTIC SIMULATORS

by

Kenneth Blake MacDonald

Thesis

submitted in partial fulfillment of
the requirements of the Degree of
Bachelor of Science with Honours.

Acadia University

May 2012

© Kenneth Blake MacDonald, 2012

This thesis by Kenneth Blake MacDonald
is accepted in its present form by the
Department of Mathematics and Statistics
as satisfying the thesis requirements for the Degree of
Bachelor of Science with Honours.

Approved by the Thesis Supervisors

_____	_____
Dr. P. Ranjan	Date

_____	_____
Dr. H. Chipman	Date

Approved by the Head of the Department

_____	_____
Dr. J. Hooper	Date

Approved by the Honours Committee

_____	_____
Dr. P. Ranjan	Date

I, Kenneth Blake MacDonald, hereby grant permission to the University Librarian at Acadia University to provide copies of the thesis, on request, on a non-profit basis.

Signature of Author

Signature of Supervisor

Signature of Supervisor

Date

Acknowledgements

First and foremost, I would like to thank my supervisors, Dr. Pritam Ranjan and Dr. Hugh Chipman, for their support over the past year. Without their patience, guidance and commitment to the project, I would not have been able to pull it all together. I would also like to thank Dr. Wilson Lu, for providing helpful comments on my thesis.

I would also like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for awarding my Undergraduate Student Research Award (USRA) in the summer of 2011, making all of this possible. I would also like to thank the professors I had, who provided me with countless learning opportunities and support throughout my undergraduate program.

To anyone I may have missed, I would like to thank you for the support and encouragement that you provided anytime during this project – Thank you!

Table of Contents

Acknowledgements	iv
Table of Contents	v
Abstract	vi
Chapter 1. Introduction	1
Chapter 2. Background	3
2.1 Gaussian process model	3
2.2 Near-singularity of R	5
Chapter 3. Methods	7
3.1 Parameterization	8
3.2 Optimization Algorithm	13
3.3 GPfit Package	15
Chapter 4. Results	18
Chapter 5. Discussion	28
Appendix A. R Package Description	30
Bibliography	31

Abstract

Gaussian process (GP) models are commonly used for emulating computationally expensive computer simulators. Fitting a GP model requires the computation of the inverse and determinant of the spatial correlation matrix, R . The near-singularity of R can cause instability in the computation of its determinant and inverse. Ranjan et al. (2011) proposed a robust approach to overcome this problem. They used a robust but somewhat slow genetic algorithm based approach for maximizing the likelihood. In this thesis we investigate several parameterizations of the likelihood for ease of optimization. We then develop a new algorithm to find the global optimum of the likelihood surface. Moreover, we developed an R package called **GPfit** that implements the proposed methodology. Finally, we use several test functions to compare the performance of our approach with a popular GP fitting package in R.

Chapter 1

Introduction

A computer simulator (or code) is often used to model a complex real physical system that would otherwise be infeasible to study due to limited resources and time. An example of such a physical system would be modeling the reactionary process that happens during a nuclear explosion. Computer simulators can still be computationally expensive to run and execute, so they are often approximated using statistical models. In this thesis we are interested in the commonly used surrogate for computer simulators, Gaussian process (GP) models. The GP model requires an experimental design of points as inputs of the computer simulator, as well as the simulator outputs at these design points.

An essential part of fitting a GP model is the computation of a spatial correlation matrix R , as well as its determinant and inverse. In some instances, when any pair of design points in the input space are close together, R becomes near-singular, and the computation of its determinant and inverse is numerically unstable. An approach to overcoming this instability is to introduce a small “nugget” parameter into the model (e.g., Sacks et al. 1989; Neal 1997; Booker et al. 1999; Santner, Williams and Notz 2003; Gramacy and Lee 2008), but choosing an appropriate value of such a parameter

is a delicate balance between stabilization and over-smoothing the model predictions. Ranjan, Haynes, and Karsten (2011) propose a computationally stable approach that introduces a lower bound on the nugget, which prevents over-smoothing and improves the accuracy of the model.

The maximum likelihood method for estimating the parameters of a GP model is a complex optimization problem, with many local optima. The method proposed in Ranjan et al. (2011) uses a genetic algorithm (GA) approach, which finds the optimum but is computationally expensive to implement. In this paper we investigate a new approach for optimizing the likelihood function. Several different parameterizations of the correlation function are investigated regarding their impact on the ease of optimization. The likelihood surface can often have multiple local optima, and thus we develop a method that allows for multiple starting locations of a gradient based search. We used a clustering based approach on a large space-filling design over the parameter space for choosing the starting locations.

This new GP fitting procedure was developed in R, and will be distributed as the `GPfit` package on CRAN. The package consists of functions for model fitting, prediction, plotting and a few other supplementary options.

The remainder of this thesis is organized as follows: we introduce the necessary background information in Chapter 2. In Chapter 3, we propose a new parameterization of the likelihood, develop a new algorithm used to optimize the likelihood function, and then present a summary of our R package. Chapter 4 presents simulation results for comparing the performance of our method with `mlegp` (Dancik 2011), a popular GP model fitting package in R that also uses a Gaussian correlation structure. Finally, Chapter 5 concludes the thesis with a few remarks.

Chapter 2

Background

The description of GP models is covered in Section 2.1. The issues involving the near-singularity of the correlation matrix R and the nugget based approach to overcome this instability are covered in Section 2.2.

2.1 Gaussian process model

In this section we briefly review the GP model, but a more detailed description of the model can be found in Santner, Williams and Notz (2003) and Rasmussen and Williams (2006). Let the i -th input and output of the computer simulator be denoted by a d -dimensional vector, $x_i = (x_{i1}, \dots, x_{id})$, and the univariate response, $y_i = y(x_i)$, respectively. The experimental design $D_0 = \{x_1, \dots, x_n\}$ is the set of n input trials. The outputs of the simulation trials are held in the n -dimensional vector $Y = y(D_0) = (y_1, \dots, y_n)'$. The simulator output, $y(x_i)$, is modeled as

$$y(x_i) = \mu + z(x_i); \quad i = 1, \dots, n,$$

where μ is the overall mean, and $z(x_i)$ is a GP with $E(z(x_i)) = 0$, $Var(z(x_i)) = \sigma^2$, and $Cov(z(x_i), z(x_j)) = \sigma^2 R_{ij}$. In general, $y(D_0)$ has a multivariate normal

distribution, $N_n(\mathbf{1}_n\mu, \Sigma)$, where $\Sigma = \sigma^2 R$, and $\mathbf{1}_n$ is a $n \times 1$ vector of all ones (see Sacks et al. 1989, and Jones et al. 1998 for details). Although there are several choices for the correlation function, we use the Gaussian correlation because of its properties such as smoothness (or differentiability in mean square sense) and popularity in other areas like machine learning (radial basis kernels) and geostatistics (kriging). For a detailed discussion on correlation functions see Stein (1999), Santner, Williams and Notz (2003), and Rasmussen and Williams (2006). The Gaussian correlation function is a special case ($p = 2$) of the power exponential correlation family,

$$R_{ij} = \prod_{k=1}^d \exp\{-\theta_k |x_{ik} - x_{jk}|^p\}, \quad \text{for all } i, j, \quad (2.1)$$

where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)$ is a vector of hyper-parameters, and p is the smoothness parameter.

We use the GP model with Gaussian correlation to predict responses at any unsampled design point x^* . Following the maximum likelihood approach, the best linear unbiased predictor at x^* is

$$\hat{y}(x^*) = \hat{\mu} + r'R^{-1}(Y - \mathbf{1}_n\hat{\mu}) = \left[\frac{(1 - r'R^{-1}\mathbf{1}_n)}{\mathbf{1}_n'R^{-1}\mathbf{1}_n} \mathbf{1}_n' + r' \right] R^{-1}Y = C'Y,$$

with mean squared error

$$\begin{aligned} s^2(x^*) &= E\left[(\hat{y}(x^*) - y(x^*))^2\right] \\ &= \sigma^2(1 - 2C'r + C'RC) \\ &= \sigma^2\left(1 - r'R^{-1}r + \frac{(1 - \mathbf{1}_n'R^{-1}r)^2}{\mathbf{1}_n'R^{-1}\mathbf{1}_n}\right), \end{aligned}$$

where $r = (r_1(x^*), \dots, r_n(x^*))$, and $r_i(x^*) = \text{corr}(z(x^*), z(x_i))$. In practice, the parameters μ , σ^2 and $\boldsymbol{\theta}$ are replaced with their respective estimates.

2.2 Near-singularity of R

Fitting a GP model to n data points using either a maximum likelihood technique or a Bayesian approach requires the repeated computation of the determinant and inverse of a $n \times n$ correlation matrix R . Although R is positive definite by definition, near-singularity (also referred to as ill-conditioning) is a common problem in fitting GP models. Neal (1997) describes the presence of near-singularity in R resulting from any pair of design points in the input space being close together.

A popular solution to overcome near-singularity is to introduce a nugget or jitter parameter, δ , in the model (e.g., Sacks et al. 1989; Neal 1997; Booker et al. 1999; Santner, Williams and Notz 2003; Gramacy and Lee 2008) that is estimated along with the other model parameters. However, adding a nugget in the model (ie. R is replaced by $R_\delta = R + \delta I$) introduces additional smoothing in the predictor and as a result the predictor is no longer an interpolator.

To minimize the unnecessary over-smoothing due to the inclusion of a nugget, Ranjan et al. (2011) introduce a lower bound on the nugget. Let $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of R , then the condition number of R in L_2 norm is $\kappa(R) = \frac{\lambda_n}{\lambda_1}$. The lower bound given by Ranjan et al. (2011) is

$$\delta_{lb} = \max \left\{ \frac{\lambda_n(\kappa(R) - e^a)}{\kappa(R)(e^a - 1)}, 0 \right\}, \quad (2.2)$$

where e^a is the threshold of $\kappa(R)$ that ensures a well conditioned R (Ranjan et al. (2011) suggest $a = 25$ for space-filling designs).

The lower bound on the nugget is only a sufficient condition and not a necessary one for $R_{\delta_{lb}}$ to be well-conditioned. For instance, a correlation matrix with 100 design points in $(0, 1)^2$ chosen using a space-filling criterion may lead to a well-conditioned R if θ is very large. If the correlation matrix is well-conditioned, R should be used

instead of $R_{\delta_{lb}}$, thus preventing the over-smoothing that occurs when a non-zero nugget is forced into the model, which is accounted for in (2.2).

It turns out that the near-singularity in R occurs less frequently as the input dimension increases. This is somewhat intuitive because the volume of the void (or unexplored region) increases exponentially with the dimension, and a space-filling design with very many points is needed to jeopardize the conditioning of the correlation matrices as dimension increases. The likelihood function used in our algorithm includes this lower bound nugget, δ_{lb} , to remove the ill-conditioning that can be present in the correlation matrix.

The upcoming chapter presents the different parameterizations investigated, as well as a description of the new optimization algorithm that is developed and implemented in our R package called **GPfit**.

Chapter 3

Methods

In order to estimate the parameters of the GP model, the maximization of the likelihood, or equivalently, the minimization of

$$-2 \log(L_{\boldsymbol{\theta}}) \propto -\log(|R_{\delta_{tb}}^{-1}|) + n \log \left[(Y - \mathbf{1}_n \hat{\mu}(\boldsymbol{\theta}))' R_{\delta_{tb}}^{-1} (Y - \mathbf{1}_n \hat{\mu}(\boldsymbol{\theta})) \right], \quad (3.1)$$

is necessary. The Gaussian correlation function ($p = 2$ in (2.1)) is used throughout this chapter. In Section 3.1 we consider several parameterizations of the scale parameter, $\boldsymbol{\theta}$, in (2.1), and its impact on the ease of optimization. After choosing a suitable parameterization of the correlation function, Section 3.2 describes a new method for finding the global minimum of the negative log-likelihood to overcome some issues in optimization such as multiple local optima. To this effect, we propose a clustering based approach to choose multiple starting values for the optimization of the likelihood. In Section 3.3, an R package, `GPfit`, implementing this algorithm is described.

3.1 Parameterization

In this section we investigate how different parameterizations of $\boldsymbol{\theta}$ affect the search for the optimum of (3.1). For illustration, we use the 1-dimensional test function $y(x) = \log(x + 0.1) + \sin(5\pi x)$ as the simulator. In each parameterization we look at the negative log-likelihood based on $n = 20$ data points selected via a space-filling maximin Latin Hypercube Design (LHD) (Stein 1987). We use the function `maximinLHS` in the R package `lhs` (Carnell 2009) for generating the design points. We discuss the impact of three different parameterizations of $\boldsymbol{\theta}$ on the likelihood function from the optimization viewpoint.

Parameterization 1:

The initial parameterization of (3.1) was via (2.1). Recall that the correlation structure is

$$R_{ij} = \exp \left(- \sum_{k=1}^d \theta_k (x_{ik} - x_{jk})^2 \right) = \prod_{k=1}^d \exp \left(- \theta_k (x_{ik} - x_{jk})^2 \right). \quad (3.2)$$

Here, values of θ_k near 0 imply high correlation or a relatively flat surface in the k^{th} coordinate and large values of θ_k imply zero correlation or a very bumpy surface in that coordinate. The optimal values of $\boldsymbol{\theta}$ are dependent on the test function and the dimensionality of the problem. In 1-dimensional cases, with scaled inputs $x \in [0, 1]$, solutions were usually found to be below $\theta = 100$. For values of θ_k close to 0, gradient based optimization methods experienced difficulty as there tended to be multiple local minima.

Under this parameterization, the likelihood surface is prone to fluctuations near $\theta_k = 0$. Figure 3.1a shows that the likelihood for the 1-dimensional test function above does not appear to be smooth for small values of θ . However, if we zoom in, Figure 3.1b shows that the likelihood function is indeed smooth and well defined for values of θ near 0. Under Parameterization 1, the range of $\boldsymbol{\theta}$ is bounded below

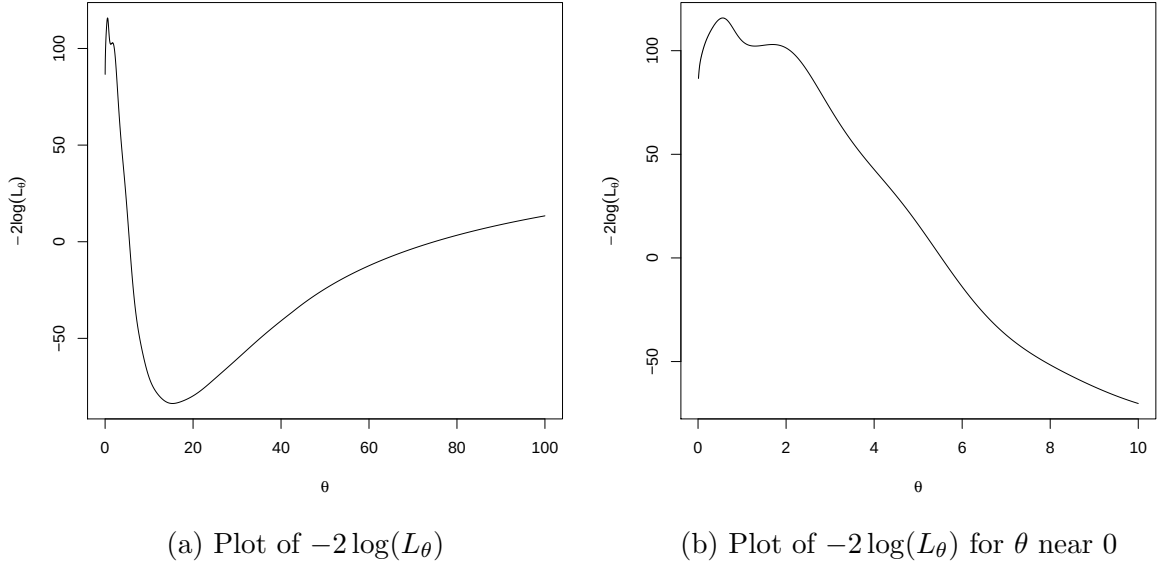


Figure 3.1: 1d illustration of $-2\log(L_\theta)$ for the simulator $y(x) = \log(x + 0.1) + \sin(5\pi x)$ under Parameterization 1 for a design size of $n = 20$.

but not above. This made the choice of starting values for finding the maximum of the likelihood difficult. Thus we consider in the next parameterization a form that bounds the correlation parameters' range from both sides.

The R package `mlegp` used for performance comparison in Chapter 4 assumes a likelihood surface with the same Gaussian correlation as used in Parameterization 1. Instead of using δ_{lb} , `mlegp` estimates the nugget parameter along with other correlation hyper-parameters.

Parameterization 2:

The second parameterization restricts the bounds of the hyper-parameter of R, by a transformation of variables to $\rho_k = \exp(-\theta_k)$. That is, the range of ρ_k is $[0, 1]$, which will hopefully allow for an easier search for the optimum of the likelihood. The new

correlation structure is now

$$R_{ij} = \prod_{k=1}^d \rho_k^{(x_{ik}-x_{jk})^2},$$

where ρ_k close to 1 implies high correlation and ρ_k close to 0 implies small correlation or a very bumpy emulator surface in the k -th coordinate.

In certain problems this parameterization did not make the minimization of $-2\log(L_{\boldsymbol{\rho}})$ any easier than Parameterization 1. The optimal values of $\boldsymbol{\rho}$ were extremely small (of the order of 10^{-15}) proving to be difficult for a gradient based search method to find, instead getting stuck at $\rho_k \approx 0$. Figure 3.2 shows the likelihood plot with respect to ρ , and the global minimum appears to be at 0. Moreover, a $\rho_k \approx 0$ would make the GP fit too wiggly, which is not necessary in this example, providing a difficult optimization for a gradient based search.

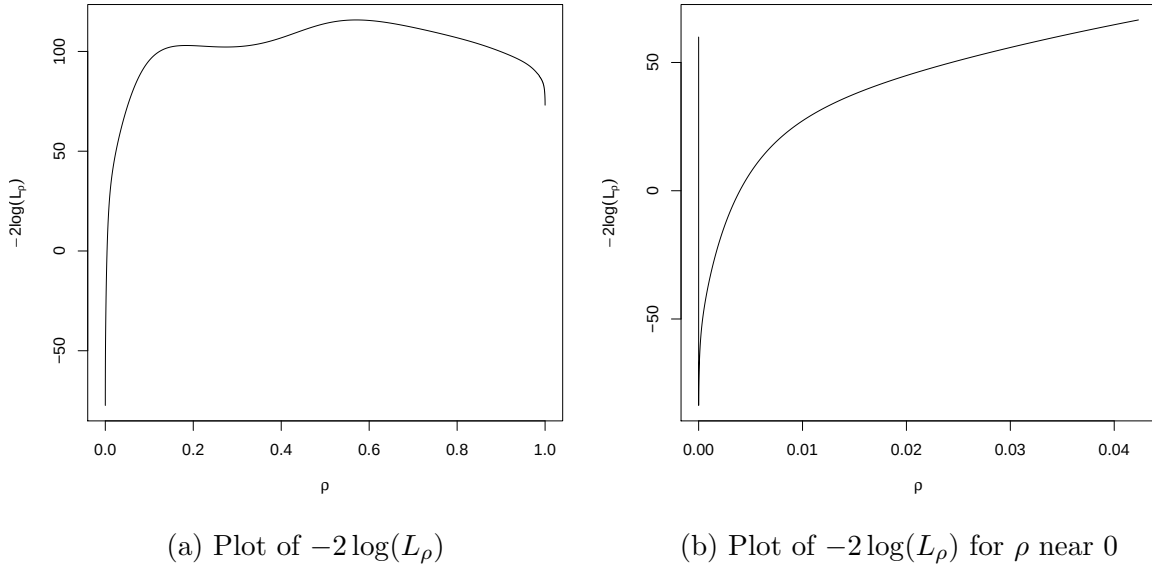


Figure 3.2: 1d illustration of $-2\log(L_{\rho})$ for the simulator $y(x) = \log(x + 0.1) + \sin(5\pi x)$ under Parameterization 2 for a design size of $n = 20$.

Although we managed to bound the parameter space, the spiky features of the likelihood surface are still present under the two parameterizations considered thus far. To this end we next consider a parameterization that is not that spiky.

Parameterization 3:

The final parameterization chosen was $\beta = \log_{10}(\theta)$. This parameter space is unbounded, however the regions with optimal parameter values have a relatively smoother likelihood surface than in Parameterization 1 (θ) and Parameterization 2 (ρ). The correlation structure under this parameterization is

$$R_{ij} = \prod_{k=1}^d \exp(-10^{\beta_k}(x_{ik} - x_{jk})^2), \quad (3.3)$$

where small values of β_k imply a very high correlation and large values of β_k imply low correlation, or a very wiggly surface in that coordinate.

Using the fact that the correlation function defined in (3.3) lies between $[0, 1]$, we can construct a range on β where optimal parameter values are likely to lie. We choose $e^{-5} = 0.0067$ and $e^{-10^{-4}} = 0.9999$ as bounds of R_{ij} as we do not expect to find a dimension that is either perfectly correlated, or perfectly uncorrelated. That is,

$$e^{-5} = 0.0067 \leq \exp\left(-\sum_{k=1}^d 10^{\beta_k}(x_{ik} - x_{jk})^2\right) \leq 0.9999 = e^{-10^{-4}},$$

or equivalently,

$$10^{-4} \leq \sum_{k=1}^d 10^{\beta_k}(x_{ik} - x_{jk})^2 \leq 5.$$

To establish ranges for β_k , we consider plausible inter-point distances, which are defined by the experimental design. For computer experiments, Loepky et al. (2009) suggest a 10 points per dimension rule. Following this rule, and assuming that $x \in [0, 1]$, the maximum value of the minimum inter-point distance is $|x_{ik} - x_{jk}| \approx \frac{1}{10}$.

Thus, the inequality can be simplified to

$$10^{-4} \leq \frac{1}{100} \sum_{k=1}^d 10^{\beta_k} \leq 5.$$

Furthermore, if we also make the assumption that $\beta_k = \beta$, for all k , that is, the simulator would be equally smooth in all directions, then the inequality becomes

$$(10^{-4})100 \leq d10^\beta \leq (5)100,$$

or,

$$-2 - \log_{10}(d) \leq \beta \leq \log_{10}(500) - \log_{10}(d). \quad (3.4)$$

We use these bounds on β for restricting the starting values to a manageable area, although solutions can be found outside this range.

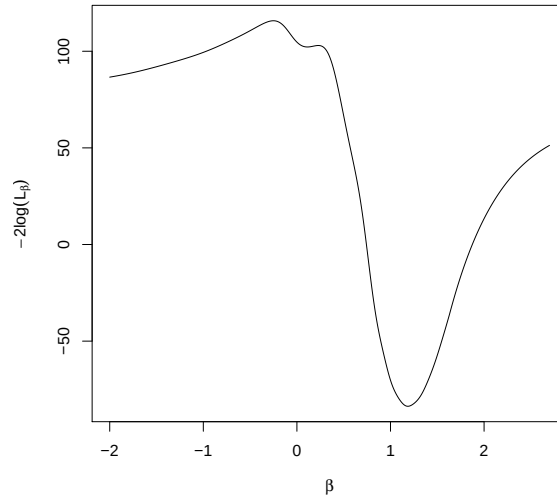


Figure 3.3: 1d illustration of $-2\log(L_\beta)$ for the simulator $y(x) = \log(x + 0.1) + \sin(5\pi x)$ under Parameterization 3 for a design size of $n = 20$ for β bounded by (3.4).

In Figure 3.3, $-2\log(L_\beta)$ is plotted as a function of β . The search area under this parameterization now has likelihood values that are smooth and well defined,

although the search area itself is unbounded. In the above 1-dimensional problem, the bounds for the initial values of β using (3.4) are $[-2, \log_{10}(500)]$. We can also see the need for several starting values for the search algorithm as there are several local minima. Depending on the problem and its dimension, the surface of the likelihood function can be quite different, and have many different local minimum values. In the next section, we discuss a new algorithm for finding the global minimum of $-2\log(L_{\beta})$ under this parameterization.

3.2 Optimization Algorithm

Ranjan et al. (2011) used a GA for finding the optimum of $-2\log(L_{\theta})$, defined in (3.1) with the correlation parameterization as in (3.2). Although the optimization is robust, the implementation of a GA is slow because of the large population sizes and the number of generations. A gradient based approach, BFGS, (Broyden, Fletcher, Goldfarb and Shanno 1970) is typically faster, but not very robust in finding the global optimum. That is, the optimization algorithm gets stuck in the local minima of $-2\log(L_{\theta})$. One can use several different starting values to improve the chances of finding the global optimum. Although the new parameterization, $\beta = \log_{10}(\theta)$, made optimization easier, the problem of multiple local optima persists. In this section we describe a new optimization algorithm which carefully chooses several starting points for the BFGS algorithm.

The algorithm to minimize $-2\log(L_{\beta})$ starts by finding a set of starting points for the BFGS algorithm. In order to have an even distribution of starting points over the range of plausible β values, we used a space-filling maximin LHD in the range specified by (3.4). The number of starting points is defined to be a function of the input dimension of the problem, and is set to (a default of) $200d$ points.

After these β values are chosen, $-2\log(L_{\beta})$ is calculated for each β and the results are stored. The next step is to take a subset of 40% (or $80d$ as default) of

the β values with the smallest $-2\log(L_\beta)$ values, and cluster them. The β values are clustered into (a default of) $2d$ groups using k-means clustering algorithm. We rerun the clustering process five times and take the best possible clustering with the smallest within group sum of squares.

Since the input variables are scaled between 0 and 1, $\beta_1 = \dots = \beta_d$ may also be plausible. It turns out that the optimal values are sometimes very close to the diagonal line, thus we also seek starting values of the BFGS algorithm on the diagonal line. A one dimensional search occurs where $\beta_k = \beta$, for all k , using the BFGS algorithm with three starting values over the range defined in (3.4). Let the bounds of the range be (a, b) , then the starting values are chosen by $\gamma(b - a) + a$, where $\gamma = 0.2, 0.5, 0.8$. The resultant β with the lowest value of $-2\log(L_\beta)$ is used as a starting point for the BFGS algorithm in addition to the other starting values described previously.

The final step of the likelihood optimization algorithm is running BFGS with the $2d + 1$ starting points that have been chosen using the clustering and diagonal search methods. The final solution is the β that corresponds to the smallest value of $-2\log(L_\beta)$ returned by BFGS. The algorithm is summarized as follows and is implemented in our new R package **GPfit** described in Section 3.3:

1. Choose a $200d$ -point maximin LHD for β in the hyper-rectangle defined by (3.4) and calculate $-2\log(L_\beta)$ for each β value.
2. Sort these $200d$ points and select the $80d$ values of β that correspond to the smallest $-2\log(L_\beta)$ values.
3. Cluster these $80d$ points into $2d$ groups using multiple k-means starts with the smallest within group sum of squares as our criteria for the best clustering, and take the centers as starting values.

4. Find the point along the diagonal $\beta_k = \beta, k = 1, \dots, d$ with the smallest $-2\log(L_\beta)$ value along the diagonal, when $d \geq 2$.
5. These $2d + 1$ (or 2 if $d = 1$) starting points, found in Steps 3 and 4, are then used in the BFGS optimization routine to find the smallest $-2\log(L_\beta)$ and corresponding β .

3.3 GPfit Package

We have developed an R package, **GPfit**, that implements the computationally stable GP fitting model (outlined in Section 2.2) with the new parameterization, (3.3), and optimization algorithm described in Section 3.1 and Section 3.2.

The variables that are used throughout the package are defined as follows: D_0 : $n \times d$ data matrix of training points, $y(D_0)$: $n \times 1$ vector of simulator responses for training points, β : $d \times 1$ correlation hyper-parameter vector, R : $n \times n$ correlation matrix, $D_{new} = \{x_1, \dots, x_{new}\}$: $n_{new} \times d$ data matrix of test points, a is the nugget threshold parameter defined in (2.2). Other parameters used by specific functions are defined as needed.

The package consists of seven functions that are accessible to the user.

`corr_matrix`

This function takes inputs of β and D_0 , and calculates the correlation matrix R , as defined in (3.2).

`likelihood_log`

This function calculates $-2\log(L_\beta)$ using a given set of training points, simulator responses, D_0 and $y(D_0)$, and hyper-parameters β . This function can take one optional parameter, a , that is used in calculating (2.2) as a nugget-threshold parameter. Although Ranjan et al. (2011) suggest $a = 25$ for space-filling designs, we set a conservative value of $a = 20$ as the default.

`GP_fit`

This function requires as inputs the training points, D_0 , and simulator responses, $y(D_0)$. There are 3 other optional parameters as well: the nugget threshold parameter a , the **control** vector that contains the 3 default values that were outlined for the algorithm in Section 3.2, as well as a **trace** flag that allows the printing of more information regarding the final BFGS runs. Using the algorithm defined in Section 3.2, this function returns the parameter estimates of the GP model (β , σ^2 , and δ_{lb}) as a class **GP** object, using `corr_matrix` and `likelihood_log` functions internally. The object of class **GP** is returned which contains D_0 , $y(D_0)$, $\hat{\beta}$, $\hat{\sigma}^2$, and $\delta_{lb}(\hat{\beta})$.

`predict.GP`

The purpose of this function is to calculate the predicted responses and the associated mean squared error, $\hat{y}(D_{new})$ and $s(D_{new})$ respectively, on a design of points D_{new} . `predict.GP` takes an object of class **GP** and D_{new} as the inputs.

`plot.GP`

The predicted response and the uncertainty, $s(D_{new})$, in $1d$ and $2d$ are plotted by this function. In $1d$ examples the training points, predicted surface, and error surfaces are depicted on one plot. In $2d$ applications two separate surface are plotted, one for each of the predicted and error surfaces. This function only requires a class **GP** object for default plotting. There are several other optional parameters that apply to the $1d$ and $2d$ plots separately. In $1d$ there are parameters that control for the plotting characters, colors, and line types for each of the variables. In $2d$ any parameters can be passed through that will be used by the `wireframe` function in the R package `lattice` (Sarkar 2008). Two more parameters control the **resolution** and **range** of the surfaces.

`summary.GP`

When prompted it returns a summary of information stored within the **GP**

object. It returns the number of observations, dimension, $\hat{\sigma}^2$, $\hat{\boldsymbol{\beta}}$, $\delta_{lb}(\hat{\boldsymbol{\beta}})$, and the nugget-threshold parameter (a) used in fitting the model.

`print.GP`

This is a shell function that uses `summary.GP` when a GP class object is called.

The package is still under final revisions and will soon be distributed on CRAN as a downloadable package. Appendix A presents the full documentation from the latest version of our `GPfit` package.

Chapter 4

Results

To test the performance of the proposed methodology and our R package, `GPfit`, we compared our results against the popular GP fitting package `mlegp`. The GP model implemented in `mlegp` fits the model defined in (3.1) under Parameterization 1 and estimates $\boldsymbol{\theta}$ and δ using a maximum likelihood method. Whereas in `GPfit`, we use the lower bound on the nugget parameter (δ_{lb}) and Parameterization 3.

The use of the lower bound on the nugget in `GPfit`, combined with a thorough optimization algorithm, should lead to better estimates in comparison to `mlegp`. The inclusion of the lower bound on the nugget resolves the erratic behavior of the likelihood surface and allows for better estimation of the hyper-parameters within the GP model when facing near-singularity.

As these two methods result in slightly different likelihood surfaces, we did not compare directly the parameter estimates of the two GP models. Instead we compared the errors between the predicted and true surfaces of several test functions (as simulators), to see which method leads to better predictions and thus finds better estimates of the global optimum of the likelihood surface.

To compare the performance of `GPfit` and `mlegp`, we use a version of the root mean squared error (RMSE) as our performance measure. First we compute

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{new}} (\hat{y}(x_i) - y(x_i))^2}{n_{new}}}$$

on the test design D_{new} for each method, then scale the RMSEs by the range of the simulator outputs, to calculate the scaled root mean squared error (SRMSE), given by

$$SRMSE = \frac{RMSE}{y_{max} - y_{min}}.$$

In each example we use the results of 50 different simulations; where we use a random $n \times d$ maximin LHD (D_0) as a training data set and a different random maximin LHD (D_{new}) of the same size for the test data in each simulation. In each instance, D_0 and D_{new} are scaled between $[0, 1]$ and are adjusted internally to the range of the function that is being used as the simulator. The average and standard error of the SRMSE on the 50 simulations is compared between `GPfit` and `mlegp` for several different design sizes. A secondary criterion used to compare methods is a count of the number of simulations, out of 50, where numerical problems caused either package to crash, although no crashes were recorded for our R package.

All of the simulations were conducted on a Dell Latitude D630, running R version 2.14.0 (2011-10-31), with the following specifications: 2.00 GB Ram, Windows Vista (32-bit), Intel(R) Core(TM)2 Duo CPU T7250 @ 2.00GHz.

Example 1

We first look at a 1-dimensional example defined by the function $y(x) = \log(x+0.1) + \sin(5\pi x)$ where $x \in [0, 1]$. Figure 4.1a shows the true function, predicted surface and the error bounds based on a space-filling design of $n = 7$ points from `GPfit`. Figure

4.1b plots $-2\log(L_\beta)$, where β is restricted to the bounds in (3.4). It is clear that the likelihood has several different local optima values, and the global minimum appears near $\beta = 2$. Figure 4.2 presents a similar plot as Figure 4.1, but for a design size of $n = 10$. As the number of training points is increased, we see better predictions as expected. Figure 4.2a presents the true function with smaller error bounds around the predicted surface. Figure 4.2b shows a similar surface for $-2\log(L_\beta)$ as in Figure 4.1b, with multiple local optima.

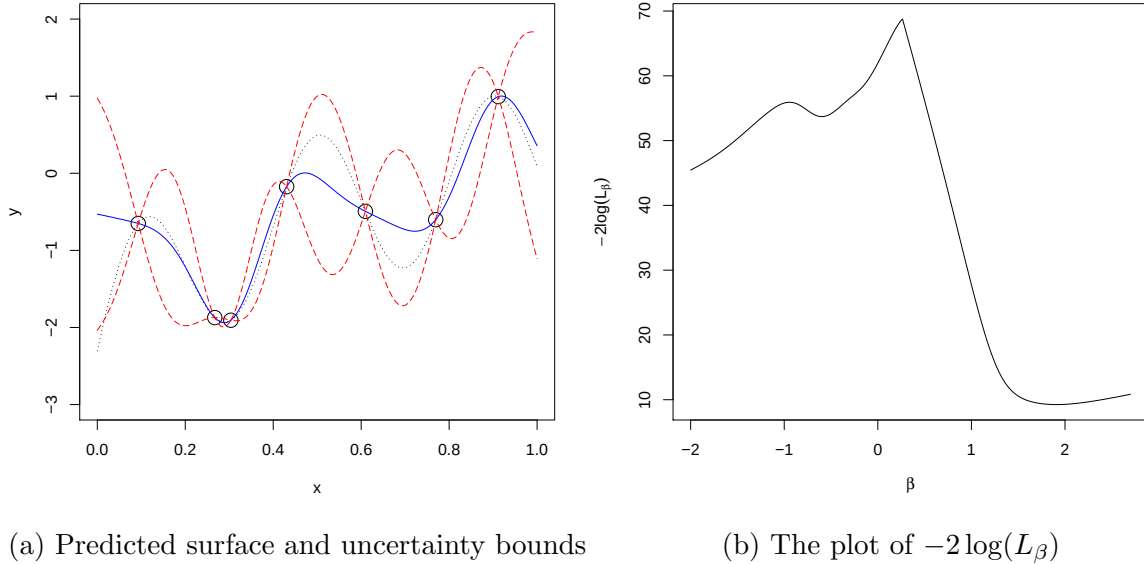
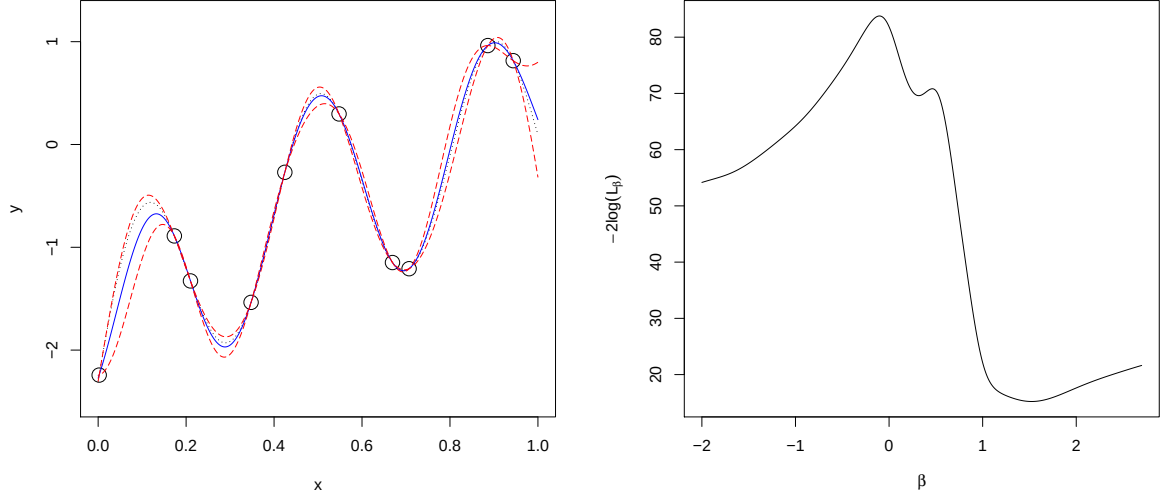


Figure 4.1: Example 1: 1d illustration of the GP fit for the test function $y(x) = \log(x + 0.1) + \sin(5\pi x)$, fitted using a 7 point maximin LHD.

In Table 4.1 we compare the performance of the two packages `GPfit` and `mlegp`, for five different values of the design size n . For each n , we summarize the average and standard errors of SRMSE over 50 simulations.

Table 4.1 shows an expected trend. For small values of n , the occurrence of near-singularity is small, so the performance of the two methods is comparable. As the value of n increases, the probability of near-singularity increases due to the close prox-



(a) Predicted surface and uncertainty bounds

(b) The plot of $-2\log(L_\beta)$

Figure 4.2: Example 1: 1d illustration of the GP fit for the test function $y(x) = \log(x + 0.1) + \sin(5\pi x)$, fitted using a 10 point maximin LHD.

imity of training points, and we see the performance of `GPfit` significantly improves over that of `mlegp`.

Table 4.1: Average (standard errors) of SRMSE for `GPfit` and `mlegp` for 50 simulations on several n , and the number of crashes that occurred for the 1d simulator $y(x) = \log(x + 0.1) + \sin(5\pi x)$.

Size	GPfit	mlegp	
	SRMSE ($\times 10^{-6}$)	SRMSE ($\times 10^{-6}$)	Crashes
$n = 10$	32958 (4948.8)	37282 (7153.4)	0
$n = 25$	139.21 (4.5498)	158.07 (15.662)	0
$n = 50$	28.81 (13.768)	113.49 (16.139)	0
$n = 75$	18.29 (1.6294)	105.84 (16.251)	0
$n = 100$	12.36 (0.7320)	101.25 (14.254)	0

Example 2

We now consider the 2-dimensional Goldstein and Price (GoldPrice) function (Andre, Siarry and Dognon 2000) for generating simulator outputs,

$$f(x_1, x_2) = \left[1 + (x_1 + x_2 + 1)^2 \{ 19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2 \} \right] * \\ \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2) \right]$$

where $x_k \in [-2, 2]$, for $k = 1, 2$, although they were scaled to $x_k \in [0, 1]$ for inputs into the optimization algorithm. A simulation study on different values of n was conducted to compare **GPfit** and **mlegp** using the SRMSE as a performance measure.

Table 4.2 presents the SRMSE values for the 50 simulations, where we used maximin LHDs of $n = 25, 50, 75$, and 100 points. As the number of points increases, the ability for **mlegp** to find numerically stable parameter estimates begins to decline due to the increased presence of near-singularity: for $n = 100$, 14 simulations failed in **mlegp**, each returning the same error message:

```
Error in solve.default(gp$invVarMatrix):  
system is computationally singular: reciprocal condition number = 2.11e-16.
```

The **mlegp** results for $n = 100$ appear to be unreliable possibly due to numerically unstable parameters estimates (i.e. crashes). This result supports the introduction of the lower bound on the nugget, as the **GPfit** method was able to find numerically stable parameter estimates under these conditions. For the other values of n , **GPfit** and **mlegp** have similar results.

Figure 4.3 displays the true and predicted GoldPrice simulator contours from both **GPfit** and **mlegp** from a space-filling maximin LHD with $n = 30$ points. This figure shows that the predictions from **GPfit** are more accurate than **mlegp** in several

Table 4.2: Average (standard errors) of SRMSE for **GPfit** and **mlegp** for 50 simulations on several n , and the number of crashes that occurred for the $2d$ GoldPrice simulator.

Size	GPfit	mlegp	
	SRMSE ($\times 10^{-4}$)	SRMSE ($\times 10^{-4}$)	Crashes
$n = 25$	381.23 (43.85)	424.07 (56.92)	0
$n = 50$	88.120 (8.114)	105.95 (18.93)	0
$n = 75$	23.282 (1.499)	17.379 (2.271)	0
$n = 100$	12.747 (0.875)	1601.5 (188.6)	14

regions, including the lower right hand corner and upper left hand corner of the plots. The predictions in the central area of the plots are also more accurate for **GPfit** then for **mlegp** as well. In this example the SRMSE values are 4.929×10^{-2} and 5.904×10^{-2} , for **GPfit** and **mlegp** respectively. In this example **GPfit** has more accurate predictions than **mlegp**.

The next few examples compare **GPfit** and **mlegp** in higher dimensions.

Example 3

In this example we consider the 4-dimensional Shekel function for generating simulator outputs, which is defined as,

$$S_{4,10}(x) = - \sum_{j=1}^{10} \left[\sum_{k=1}^4 (x_k - C_{jk})^2 + \xi_j \right]^{-1}$$

$$\xi = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)$$

Table 4.3: Average (standard errors) of SRMSE for **GPfit** and **mlegp** for 50 simulations on several n , and the number of crashes that occurred for the 4d Shekel simulator.

Size	GPfit	mlegp	
	SRMSE ($\times 10^{-3}$)	SRMSE ($\times 10^{-3}$)	Crashes
$n = 25$	52.656 (2.735)	56.078 (3.885)	0
$n = 50$	35.637 (1.881)	36.535 (2.102)	0
$n = 75$	37.347 (1.959)	37.456 (2.055)	0
$n = 100$	31.749 (1.470)	31.952 (1.549)	0

mlegp as $\delta_{lb}(\beta) = 0$. The next example shows an instance in higher dimensions where near-singularity is present.

Example 4

We next consider the 4-dimensional Colville function, which we use to generate simulator outputs from

$$\begin{aligned}
C_4(x) = & 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 \\
& + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8x_2^{-1}(x_4 - 1),
\end{aligned}$$

where $x_k \in [-10, 10]$, for $k = 1, \dots, 4$ and is scaled to $[0, 1]$ for the optimization algorithms.

Table 4.4 presents the results from the 50 simulations of several values of the design size n . For each value of n , **GPfit** outperforms **mlegp** according to SRMSE. Due to near-singularity, **mlegp** crashed several times for each of $n = 50, 75$, and 100, with the same error that was returned in Example 2:

```
Error in solve.default(gp$invVarMatrix) :
```

```
system is computationally singular: reciprocal condition number = 2.11e-16.
```

The successful completions of `mlegp` for these design sizes also found poor solutions, evident from the large values for the average and standard error of SRMSE. For each value of n , `GPfit` consistently found numerically stable and better fits for the simulator surface as the SRMSE values decrease as the value of n increases.

Table 4.4: Average (standard errors) of SRMSE for `GPfit` and `mlegp` for 50 simulations on several n , and the number of crashes that occurred for the 4d Colville simulator.

Size	GPfit	mlegp	
	SRMSE ($\times 10^{-6}$)	SRMSE ($\times 10^{-6}$)	Crashes
$n = 25$	103.3 (5.401)	109.58 (6.120)	0
$n = 50$	11.77 (0.771)	10334 (3344)	2
$n = 75$	7.169 (0.472)	3251 (1109)	5
$n = 100$	5.786 (1.839)	63.10 (25.39)	1

Example 5

We now consider the 6-dimensional Hartmann function as the simulator,

$$H_{6,4}(x) = - \sum_{j=1}^4 \alpha_j \exp \left[- \sum_{k=1}^6 B_{jk} (x_k - Q_{jk})^2 \right]$$

$$\alpha = (1, 1.2, 3, 3.2)$$

$$B = \begin{pmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

defined for $x_k \in [0, 1]$ for $k = 1, \dots, 6$.

Table 4.5 presents the results of the simulations for five different values of the design size n . We see a similar trend as in Example 3 where the values of SRMSE are comparable between `GPfit` and `mlegp`. The results in Table 4.5 once again show that when there is no presence of near-singularity in the correlation matrix R , the `GPfit` method involving the lower bound of the nugget has an equivalent performance to `mlegp`.

Table 4.5: Average (standard errors) of SRMSE for `GPfit` and `mlegp` for 50 simulations on several n , and the number of crashes that occurred for the *6d* Hartmann simulator.

Size	GPfit	mlegp	
	SRMSE ($\times 10^{-3}$)	SRMSE ($\times 10^{-3}$)	Crashes
$n = 25$	118.44 (4.837)	116.64 (4.655)	0
$n = 50$	105.24 (4.649)	105.56 (4.500)	0
$n = 75$	82.587 (2.536)	84.819 (3.090)	0
$n = 100$	75.169 (2.645)	75.402 (2.738)	0
$n = 125$	63.014 (1.652)	63.223 (1.653)	0

Chapter 5

Discussion

This thesis explored different parameterizations of the likelihood and Gaussian correlation functions in order to make the optimization more robust. The final parameterization, $\beta = \log_{10}(\theta)$, is described under Parameterization 3 in Section 3.1.

After finding a suitable parameterization of the model, the development of the optimization algorithm for finding the global minimum of $-2\log(L_\beta)$ is described in Section 3.2. This algorithm combines several different starting values determined by a clustering process, with a gradient based search approach to find the global minimum. The lower bound nugget method outlined in Ranjan et al. (2011), along with our new parameterization, found better optima of the likelihood function despite the presence of near-singularity in R .

Of the 1100 simulations we conducted for each package, there were 22 crashes within the `mlegp` package. These crashes were the result of near-singularity in R , and the performance of simulations of similar design sizes also suffered when `mlegp` found a solution instead of crashing. The prediction errors from `mlegp` summarized over the successful fits in the cases with a few crashes ($n = 100$ in Example 2 and

$n = 50, 75$, and 100 in Example 4) are usually large and hence unreliable. Of the 1100 simulations for `GPfit`, there were no crashes.

When R is well-behaved, the method of choosing the lower bound on the nugget reverts back to the original method as the lower bound is set equal to 0. This makes the performance of the two packages comparable as seen in Examples 3 and 5.

It was noted that the speed of `mlegp` is faster than `GPfit` when fitting the GP models. The superior performance in speed was the result of `mlegp` being written in pre-compiled C code, whereas the our method was implemented solely in R. Further work can be done to improve the speed of our package by implementing the process through C code. We also intend to implement the iterative approach proposed by Ranjan et al. (2011). Including this feature into `GPfit` will make the package more useful to practitioners.

Appendix A

R Package Description

GPfit package manual....

Bibliography

- [1] Ababou, R., Bagtzoglou, A. C. and Wood, E. F. (1994). On the condition number of covariance matrices in kriging, estimation, and simulation of random fields. *Mathematical Geology*, **26**, 99-133.
- [2] Andre, J., Siarry, P. and Dognon, T. (2000). An improvement of the standard genetic algorithm fighting premature convergence. *Advances in Engineering Software*, **32**, 49-60.
- [3] Barton, R. R. and Salagame, R. R. (1997). Factorial hypercube designs for spatial correlation regression. *Journal of Applied Statistics*, **24**, 453-473.
- [4] Booker, A. J., Dennis Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V. and Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, **17**, 1-13.
- [5] Booker, A. (2000). *Well-conditioned Kriging models for optimization of computer simulations*. Mathematics and Computing Technology Phantom Works - The Boeing Company, M&CTTECH-00-002.
- [6] Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and its Applications*, **6**, 76-90.
- [7] Carnell, R. (2009). *lhs: Latin Hypercube Samples*. R package version 0.5.
- [8] Dancik, G. M. (2011). *mlegp: Maximum Likelihood Estimates of Gaussian Processes*. R package version 3.1.2. <http://CRAN.R-project.org/package=mlegp>
- [9] Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, **13**(3), 317-322.

- [10] Goldfarb, D. (1970). A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, **24**(109), 23-26.
- [11] Gramacy, R. B. and Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, **103**(483), 1119-1130.
- [12] Jones, D., Schonlau, M. and Welch, W. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, **13**, 455-492.
- [13] Loeppky, J. L., Sacks, J. and Welch W. J. (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, **51**(4), 366-376.
- [14] Neal, R. M. (1997). *Monte Carlo implementation of Gaussian process models for Bayesian regression and classification*. Tech Rep. 9702, Dept. of Statistics, Univ. of Toronto, Canada.
- [15] Ranjan, P., Haynes, R. and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data. *Technometrics*, **53**(4), 366-378.
- [16] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.
- [17] Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, **4**, 409-423.
- [18] Santner, T. J., Williams, B. J. and Notz, W. (2003). *The Design and Analysis of Computer Experiments*. Springer-Verlag Inc., New York, NY.
- [19] Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer, New York. ISBN 978-0-387-75968-5.

- [20] Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, **24**, 647-656.
- [21] Stein, M. (1987). Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics*, **29**, 143-151.
- [22] Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, NY.