Safety, Reliability and Risk Analysis: Theory, Methods and Applications – Martorell et al. (eds)
© 2009 Taylor & Francis Group, London, ISBN 978-0-415-48513-5

Numerical study of algorithms for metamodel construction and validation

Bertrand Iooss, Loic Boussouf & Amandine Marrel
CEA Cadarache, Saint Paul lez Durance, France

Vincent Feuillard
EADS CCR, Suresnes, France

ABSTRACT: In some studies requiring predictive numerical models, it can be advantageous to replace cpu time expensive computer models by cpu-inexpensive mathematical functions, called metamodels. In this paper, we focus on the Gaussian process metamodel whose construction requires an initial design of computer model simulations. A numerical study compares different types of Latin hypercubes maximized relatively of metamodel predictivity. The metamodel validation step consists in estimating the true metamodel predictivity with a minimum number of additional calculations. In this goal, we propose and test an algorithm which optimizes the distance between the test points and points of the initial training base.

1 INTRODUCTION

With the advent of computing technology and numerical methods, investigation of computer code experiments remains an important challenge. Complex computer models calculate several output values (scalars or functions) which can depend on a high number of input parameters and physical variables. These computer models are used to make simulations as well as predictions, uncertainty analyses or sensitivity studies (De Rocquigny, Devictor, and Tarantola pear).

However, complex computer codes are often too time expensive to be directly used to conduct uncertainty propagation studies or global sensitivity analysis based on Monte Carlo methods. To avoid the problem of huge calculation time, it can be useful to replace the complex computer code by a mathematical approximation, called a response surface or a surrogate model or also a metamodel. The response surface method consists in constructing a function from few experiments, that simulates the behavior of the real phenomenon in the domain of influential parameters. These methods have been generalized to develop surrogates for costly computer codes (Sacks, Welch, Mitchell, and Wynn 1989; Kleijnen and Sargent 2000). Several metamodels are classically used: polynomials, splines, generalized linear models, or learning statistical models like neural networks, regression trees, support vector machines (Fang, Li, and Sudjianto 2006).

In this work, we pay attention to one particular class of metamodels, the Gaussian process model which extends the kriging principles of geostatistics (Matheron 1970) to computer experiments by considering the correlation between two responses of a computer code depending on the distance between input variables (Sacks, Welch, Mitchell, and Wynn 1989). The Gaussian process model presents several advantages, especially the interpolation and interpretability properties. Moreover, numerous studies have shown that this model can provide a statistical framework to compute an efficient predictor of code response (Santner, Williams, and Notz 2003).

From a practical standpoint, constructing a Gaussian process model implies estimation of several hyperparameters included in the covariance function. This optimization problem is particularly difficult for a model with many inputs and inadequate sampling designs (Fang, Li, and Sudjianto 2006; Marrel, Iooss, Van Dorpe, and Volkova 2008). Several authors (for example Simpson, Peplinski, Kock, and Allen 2001) have shown that the space filling designs are well suited to metamodel fitting. However, this class of design, which aims at obtaining the better coverage of the points in the space of the input variables, is particularly large, ranging from the well known Latin Hypercube Samples to low discrepancy sequences (Fang, Li, and Sudjianto 2006). No theoretical result gives the type of initial design which leads to the best fitted Gaussian process metamodel in terms of metamodel predictivity. In this work, we propose to give some numerical results in order to answer to this fundamental question. Another important issue we propose to adress concerns the optimal choice of the

test basis, i.e. the test basis which allows the most accurate metamodel validation using the minimal number of additional test observations.

In the following section, we present the Gaussian process model. In the third section, we present several criteria to optimize the initial input design. On an analytical example, we evaluate the numerical performance of the optimal design in terms of predictivity of a fitted Gaussian process metamodel. In the fourth section, we look at the metamodel validation problem. Our solution consists in minimizing the number of test observations by using the recent algorithm of Feuillard 2007. Numerical results show the high efficiency of this algorithm compared to a simple Monte-Carlo strategy. Finally, a conclusion gives some perspectives of this work.

2 GAUSSIAN PROCESS MODELING

Let us consider n realizations of a computer code. Each realization $y(\boldsymbol{x}) \in \mathbb{R}$ of the computer code output corresponds to a d-dimensional input vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \chi$, where χ is a bounded domain of $\mathbb{R}^d$. The n points corresponding to the code runs are called the experimental design and are denoted as $X_s = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)})$. The outputs will be denoted as $Y_s = (y^{(1)}, \ldots, y^{(n)})$ with $y^{(i)} = y(\boldsymbol{x}^{(i)}) \ \forall \ i = 1..n$. Gaussian process (Gp) modeling treats the deterministic response $y(\boldsymbol{x})$ as a realization of a random function $Y(\boldsymbol{x})$, including a regression part and a centered stochastic process. This model can be written as:

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + Z(\boldsymbol{x}).$$

The deterministic function $f(\boldsymbol{x})$ provides the mean approximation of the computer code. Our study is limited to the one-degree polynomial regression where $f(\boldsymbol{x})$ can be written as follows:

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j,$$

where $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_k]^t$ is the regression parameter vector.

The stochastic part $Z(\boldsymbol{x})$ is a Gaussian centered process fully characterized by its covariance function: $\mathrm{Cov}(Z(\boldsymbol{x}), Z(\boldsymbol{u})) = \sigma^2 R(\boldsymbol{x}, \boldsymbol{u})$, where σ^2 denotes the variance of Z and R is the correlation function that provides interpolation and spatial correlation properties. To simplify, a stationary process $Z(\boldsymbol{x})$ is considered, which means that correlation between $Z(\boldsymbol{x})$ and $Z(\boldsymbol{u})$ is a function of the distance between $\boldsymbol{x}$ and $\boldsymbol{u}$. Our study is focused on a particular family of correlation functions that can be written as a product of one-dimensional correlation functions R_l:

$$\mathrm{Cov}(Z(\boldsymbol{x}), Z(\boldsymbol{u})) = \sigma^2 R(\boldsymbol{x} - \boldsymbol{u}) = \sigma^2 \prod_{l=1}^{d} R_l(x_l - u_l).$$

We use the generalized exponential correlation function:

$$R_{\theta,p}(\boldsymbol{x} - \boldsymbol{u}) = \prod_{l=1}^{d} \exp(-\theta_l |x_l - u_l|^{p_l}),$$

where $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_d]^t$ and $\boldsymbol{p} = [p_1, \ldots, p_d]^t$ are the correlation parameters with $\theta_l \geq 0$ and $0 < p_l \leq 2$ $\forall l = 1..d$.

If a new point $\boldsymbol{x}^* = (x_1^*, \ldots, x_d^*) \in \chi$ is considered, we obtain:

$$\mathbb{E}[Y_{\mathrm{Gp}}(\boldsymbol{x}^*)] = f(\boldsymbol{x}^*) + \boldsymbol{k}(\boldsymbol{x}^*)^t \Sigma_s^{-1} (Y_s - f(X_s)), \quad (1)$$

$$\mathrm{Var}[Y_{\mathrm{Gp}}(\boldsymbol{x}^*)] = \sigma^2 - \boldsymbol{k}(\boldsymbol{x}^*)^t \Sigma_s^{-1} \boldsymbol{k}(\boldsymbol{x}^*), \quad (2)$$

with $Y_{\mathrm{Gp}} \sim (Y | Y_s, X_s, \boldsymbol{\beta}, \sigma, \boldsymbol{\theta}, \boldsymbol{p})$,

$$\begin{aligned} \boldsymbol{k}(\boldsymbol{x}^*) &= [\mathrm{Cov}(y^{(1)}, Y(\boldsymbol{x}^*)), \ldots, \mathrm{Cov}(y^{(n)}, Y(\boldsymbol{x}^*))]^t \\ &= \sigma^2 [R_{\boldsymbol{\theta},\boldsymbol{p}}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^*), \ldots, R_{\boldsymbol{\theta},\boldsymbol{p}}(\boldsymbol{x}^{(n)}, \boldsymbol{k}^*))]^t \end{aligned}$$

and the covariance matrix

$$\Sigma_s = \sigma^2 (R_{\boldsymbol{\theta},\boldsymbol{p}}(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)})_{i=1..n, j=1..n}).$$

The conditional mean (Eq. (1)) is used as a predictor. The variance formula (Eq. (2) corresponds to the mean squared error (MSE) of this predictor and is also known as the kriging variance. This analytical formula for MSE gives a local indicator of the prediction accuracy. More generally, Gp model provides an analytical formula for the distribution of the output variable at an arbitrary new point. This distribution formula can be used for sensitivity and uncertainty analysis, as well as for quantile evaluation (O'Hagan 2006).

Regression and correlation parameters $\beta, \sigma, \boldsymbol{\theta}$ and $\boldsymbol{p}$ are ordinarily obtained by maximizing likelihood functions (Fang, Li, and Sudjianto 2006). This optimization problem can be badly conditioned and difficult to solve in high dimensional cases ($d > 10$) (Marrel, Iooss, Van Dorpe, and Volkova 2008). Moreover, the estimation algorithms are particularly sensitive to the input design. The following section proposes to deal with the input design problem.

3 INITIAL DESIGN FOR THE METAMODEL CONSTRUCTION

For computer experiments, selecting an experimental design is a key issue in building an efficient and informative metamodel. For example, a well known Latin Hypercube Sample (LHS) $(X^{(1)}, \ldots, X^{(n)})$ of a random variable X gives a sample mean $m = \frac{1}{n}\sum_{i=1}^{n} X^{(i)}$ with smaller variance than the sample mean of a Simple Random Sample (SRS). Contrary to the SRS which consists of n independent identical samples, the LHS consists in dividing the domain of each input variable in n equiprobable strata, and sample once from each stratum.

However, LHS does not reach the smallest possible variance for the sample mean. Since it is only a form of stratified random sampling and is not directly related to any criterion, it may also perform poorly in estimation and prediction of the response at untried sites. Therefore, some authors have proposed to enhance LHS not only to fill space in one dimensional projection, but also in higher dimensions (Park 1993). One powerful idea is to adopt some optimality criterion for construction of LHS, such as entropy, integrated mean square error, minimax and maximin distances, etc. The maximin criterion is a popular metamodel-independent criterion which consists in maximizing the minimal distance between the points. This leads to avoid situations with too close points.

Alternative metamodel-independent criteria, based on discrepancy measures, consist in judging the uniformity quality of the design. Discrepancy can be seen as a measure between an initial configuration and an uniform configuration. It is a comparison between the volume of intervals and the points within these intervals (Hickernell 1998). There exists different kinds of definition using different forms of intervals or different norms in the functional space. Two measures have shown remarkable properties (Jin, Chen, and Sudjianto 2005; Fang, Li, and Sudjianto 2006):

- the centered L^2 discrepancy

$$D^2(X_s(n)) = \left(\frac{13}{12}\right)^d - \frac{2}{n}\sum_{i=1}^{n}\prod_{k=1}^{d} \times \left(1 + \frac{1}{2}|u_k^{(i)} - \frac{1}{2}| - \frac{1}{2}|u_k^{(i)} - \frac{1}{2}|^2\right) + \frac{1}{n^2}\sum_{i,j=1}^{n}\prod_{k=1}^{d}\left(1 + \frac{1}{2}|u_k^{(i)} - \frac{1}{2}| + \frac{1}{2}|u_k^{(j)} - \frac{1}{2}| - \frac{1}{2}|u_k^{(i)} - u_k^{(j)}|\right) \quad (3)$$

  where $X_s(n)$ denotes the input learning sample with n input vectors and $\left(u_k^{(i)}\right)_{i=1..n,k=1..d}$ are the normalized values in $[0,1]$ of the design $X_s(n) = \left(x_k^{(i)}\right)_{i=1..n,k=1..d}$;

- the wrap-around L^2 discrepancy

$$W^2(X_s(n)) = \left(\frac{4}{3}\right)^d + \frac{1}{n^2}\sum_{i,j=1}^{n}\prod_{k=1}^{d}\left[\frac{3}{2} - |u_k^{(i)} - u_k^{(j)}|(1 - |u_k^{(i)} - u_k^{(j)}|)\right] \quad (4)$$

  which allows to suppress bound effects (by wrapping the unit cube for each coordinate).

The optimization of LHS can be done following different methods: choice of the best (in terms of the chosen criteria) LHS amongst a large number of different LHS, columnwise-pairwise algorithms, genetic algorithms, simulated annealing, etc (see for example Liefvendahl and Stocki 2006). In our tests, we have found that the simulated annealing algorithm with geometrical descent gives the best results for all the criteria. Figure 1 gives some example of two-dimensional LHS of size $n = 16$ optimized following three different criteria with the simulated annealing algorithm. We see that uniform repartitions of the points are nicely respected.

At present, we perform a numerical study to evaluate the impact of an inadequate design on the metamodel fitting process. For the metamodel, we use the Gaussian process model Y_{Gp} described in §2. The quality of the metamodel predictor is measured

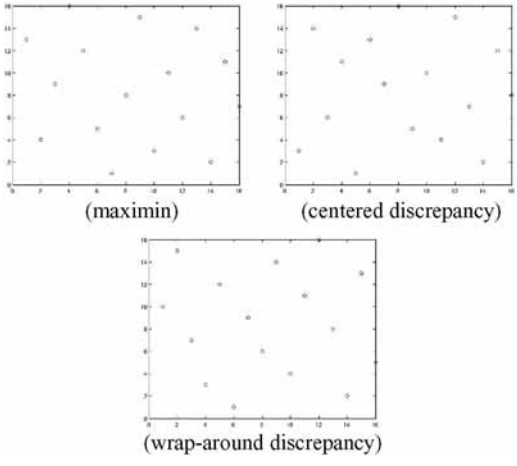

Figure 1. Visual comparison of LHS optimized following different criteria ($d = 2, n = 16$).

by the so-called predictivity coefficient Q_2 (i.e. the determination coefficient R^2 computed on a test basis) which gives the percentage of output values explained by the metamodel:

$$Q_2 = 1 - \frac{\sum_{i=1}^{n_t}[y(\boldsymbol{x}_t^{(i)}) - \hat{Y}_{\mathrm{Gp}}(\boldsymbol{x}_t^{(i)})]^2}{\sum_{i=1}^{n_t}[\bar{y} - y(\boldsymbol{x}_t^{(i)})]^2} \tag{5}$$

with $(\boldsymbol{x}_t^{(1)}, \ldots, \boldsymbol{x}_t^{(n_t)})$ the test sample of size n_t, $\hat{Y}_{\mathrm{Gp}} = \mathbb{E}(Y_{\mathrm{Gp}})$ the Gp predictor (Eq. (1)) and $\bar{y}$ the mean of the output test sample $(y(\boldsymbol{x}_t^{(1)}), \ldots, y(\boldsymbol{x}_t^{(1)}))$.

Our tests involve a five-dimensional analytical function (called the g-Sobol 5d function):

$$f(\boldsymbol{x}) = \sum_{i=1}^{5} \frac{|4x_i - 2| + a_i}{1 + a_i}$$

with $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 4$, $a_5 = 5$, $x \in [0,1]^5$. We have made several comparisons to study the different space filling designs before fitting a metamodel. For a size n of the learning sample and each type of design, we repeat 100 times the following procedure: we generate an initial design of n observations, we fit a Gp metamodel and evaluate its predictivity coefficient Q_2 using a large test sample. Therefore, we obtain 100 values of Q_2 whose mean and variance give us the efficiency and robustness of the design in terms of Gp quality.

The initial design LHS optimized with the wrap-around discrepancy (Eq. (4)) has given us the best results. In Figure 2, we compare the predictivity coefficients obtained with non optimized LHS and those obtained with LHS optimized with the wrap-around discrepancy. The size of the design increases from $n = 22$ to $n = 40$, which leads to a regular increase of Q_2. For each size n, the boxplot represents the summary of the 100 values of Q_2. In the all range of n, Q_2 of the wrap-around optimized LHS (called WLHS) are better than the random LHS, with much smaller variances. For small sample sizes, the Q_2 differences

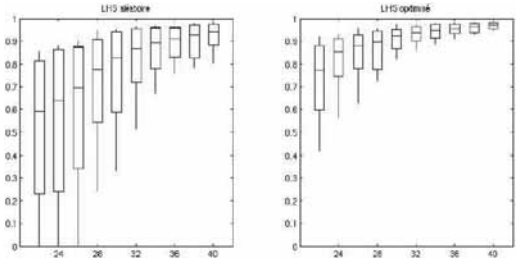

Figure 2. For the g-Sobol 5d function, Gp Q_2 evolution in function of the learning sample size n and for two types of LHS (left: random LHS; right: wrap-around opimized LHS).

reach 0.2 (Q_2(LHS) $\sim$ 0.6 and Q_2(WLHS) $\sim$ 0.8). In industrial applications, such a difference makes the distinction between "bad" (unacceptable) metamodels and good ones. The latter can be used for example for quantitative sensitivity studies.

Furthermore, we have found that the LHS optimized with the wrap-around discrepancy guarantees correct repartitions of the points for all the two-dimensional projections, while other types of LHS (like maximin) have bad repartitions for these projections. All the LHS guarantee good repartitions for one-dimensional projections, but not for the other dimensions of projection. Our tests have been performed for dimensions $d = 3, 4, 5, 10, 15$. This property is particularly important when the initial design is made in dimension d and the metamodel construction is made in a smaller dimension. In practice, this is often the case because the initial design may reveal with screening methods the useless (not influent) input variables that we can neglect during the metamodel construction step.

In conclusion of our numerical study, the LHS optimized with the wrap-around discrepancy has shown excellent results for the Gp metamodel construction, even in high dimension. Other types of LHS can also provide good results but less systematically.

4 TEST BASIS SELECTION FOR METAMODEL VALIDATION

The validation of a metamodel is a problem often neglected, in theory and in practice. Two methods are ordinarily used: the test basis and the cross validation method. The first consists in comparing the metamodel predictions on obervation points not used in the fitting process. This gives some prediction residuals (which can be finely analyzed) and global quality measures as the predictivity coefficient Q_2 (Eq. (5)). Such test points set is called a test basis (or also validation basis or prediction basis). This method requires new calculations with the computer code and the first question we have to face up is the right number of prediction points to obtain required accuracy of our global validation measures. For cpu time consuming computer code, it can be difficult to provide sufficient test points. Some convergence visualisation tools of the global validation measures can be used to answer to this first question. Another important question is the localization of these test points. The usual practice is to choose an independent Monte-Carlo sample for the test basis. However, if the sample size is small, we claim that the proposed points can be badly localized, for example near learning points or leaving large space domain unsampled. As in the previous paragraph, a fine strategy could be to use a space filling design as the test sample. Unfortunately, this solution does not

avoid the possibility of too strong proximity between learning and test points, which leads to too optimistic quality measures.

The second solution to validate a metamodel, which is extremely popular in practice, avoids new calculations on the computer code. The cross validation method (and its alternative, the bootstrap method) proposes to divide the initial sample on a learning sample and a test sample. A metamodel is estimated with the points in the new learning sample and prediction residuals are obtained via the new test sample. This process is repeated several times by using other divisions of the learning sample. Finally all the prediction residuals can be used to compute the global predictivity measures. The first drawback of this method is its cost, which can become large due to many metamodel fitting processes. Moreover, if the initial design has a special optimized structure, the points selection step causes the breakdown of this structure for the new learning sample. The nice properties of this learning sample are no more present and the metamodel fitting process might fail for each different iteration of the cross validation process. This leads to too pessimistic quality measures.

To sum up, the test basis method requires too many new prediction points while the cross-validation method can provide too pessimistic validation criteria. Therefore, we propose to introduce a new solution to solve this dilemma. We retain the first solution, which consists in creating a new test basis, but want to limit its main drawback by minimizing the number of new prediction points which are required. In this goal, we retain a recent algorithm developed by Feuillard 2007, which allows the specification of new design points decreasing the discrepancy of an initial design. This sequential algorithm gives us at each step the prediction point furthest away from the other points of the design. The algorithm performs its optimization process in the space χ of the input variables $\boldsymbol{x}$, without taking into account the ouput values Y of the learning sample design. We hope that, by choosing the future prediction points in the unfilled zone of the learning sample design, we capture the right metamodel predictivity using only a small number of additional points.

Let us consider $X_f(n_f) = (\boldsymbol{x}_f^{(i)})_{i=1..n_f}$ a low discrepancy sequence of n_f points in $[0,1]^d$. A low discrepancy sequence is a deterministic design which has a smaller discrepancy than a random design. It is constructed with special algorithms to uniformly fill the space with regular patterns. Among all the low discrepancy sequence, Halton, Hammersley, Faure and Sobol sequences are the most famous. In the following, we will use the Hammersley sequence which, on a few tests, have shown better properties than the others (Feuillard 2007). The chosen discrepancy measure is the centered L^2 discrepancy D (Eq. (3)).

To obtain a new point to the initial n-size sample, noticed $X(n)$, we use the following algorithm:

1. For $i = 1, \ldots, n_f$,
   - $X(n+1) = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \cup \boldsymbol{x}_f{}^{(i)}$;
   - calculation of $\mathrm{Dif}_i = D(X(n+1)) - D(X(n))$;
2. selection of i^* such that $\mathrm{Dif}_{i^*} = \min_{i=1,\ldots,n_f} \mathrm{Dif}_i$;
3. we obtain the new point $\boldsymbol{x}_f{}^{(i^*)}$.

This algorithm can be repeated sequentially to obtain n_t points, by updating the initial design and the low discrepancy sequence. For example, for the second point, we reinitalize the design by the following: $X(n+1) = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \cup \boldsymbol{x}_f{}^{(i^*)}$ and $X_f(n_f - 1) = \{x_f{}^{(1)}, \ldots, x_f{}^{(n_f)}\} \backslash x_f{}^{(i^*)}$. This algorithm just consists in adding to the initial design some points of a low discrepancy sequence by minimizing the discrepancies difference between the initial and the new design. The size of the low discrepancy sequence is required to be as large as possible, especially if d is large. Figure 3 gives an example of the specification with our algorithm of $n_t = 4$ new points (the crosses) inside an initial design of $n = 46$ points (Monte-Carlo sample, $d = 2$) One of the advantage of this algorithm is its size-independence (related to the number of added points): the sequence of added points is deterministic and will be always the same for the same $X_f(n_f)$.

To illustrate the advantage of using this algorithm for metamodel validation purpose, we perform an anlytical test using an eight-dimensional analytical

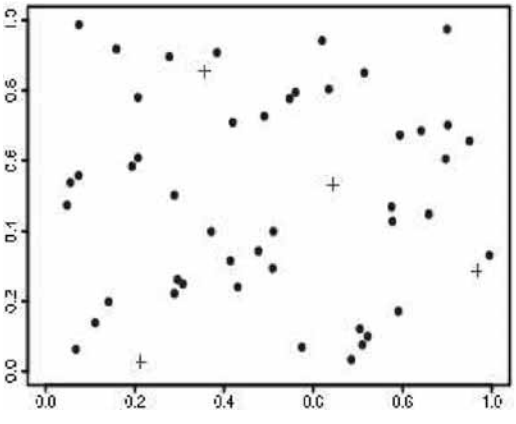

Figure 3. Example of the sequential algorithm: $n = 46$, $d = 2$, $n_t = 4$. The bullets are the points of the initial design while the crosses are the new specified points.

function (called the g-Sobol 8d function):

$$f(\boldsymbol{x}) = \sum_{i=1}^{8} \frac{|4x_i - 2| + a_i}{1 + a_i}$$

with $a_1 = a_2 = 3$, $a_i = 0$ for $(i = 3, \ldots, 8)$, $\boldsymbol{x} \in [0,1]^8$. The learning sample is a maximin optimized LHS of size $n = 40$ (see §3). For the metamodel, we use the Gaussian process model Y_{Gp} described in §2. We compute the reference predictivity coefficient by mean of 100 test samples of size $n_t = 1000$ and obtain $Q_2^{\text{ref}} = 0.83$. We then apply the sequential algorithm described previously (with a Hammersley sequence of size $n_f = 10000$) by adding $n_t = 50$ new points to the design, and we obtain $Q_2^{\text{seq50}} = 0.85$, which is close to the true value. We compare this result with 100 simple Monte-Carlo samples of the same size ($n_t = 50$) which give the 90% confidence interval $[0.79, 0.91]$ for Q_2^{MC}. This last result is rather large and shows the insufficient number of points if we choose a simple Monte-Carlo design. Moreover, it shows too optimistic results compared to the reference value: the interval $[0.79, 0.91]$ is not centered on the value 0.83.

Figure 4 shows the evolution of the obtained Q_2 for test bases with different sizes, ranging from $n_t = 10$ to $n_t = 50$. The solid line shows the results obtained with the deterministic sequential design explained previously, while the dotted lines show the 100 sequentially

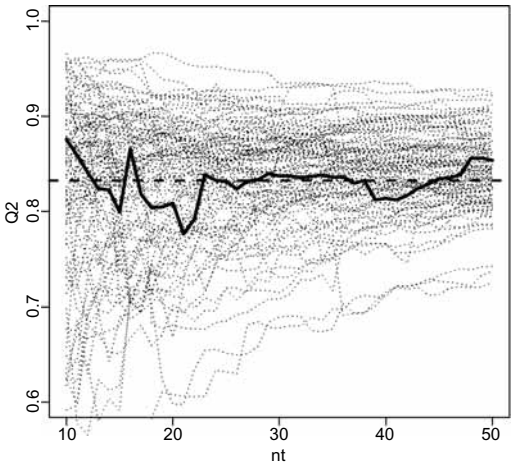

Figure 4. For the g-Sobol 8d function, Q_2 evolution (of a fitted Gp) in function of the test basis size n_t and for two types of validation design: the plain line corresponds to the Feuillard sequential design and the dotted lines correspond to 100 different Monte-Carlo samples. The dashed lines is the Q_2 reference value (mean of 100 test samples of size $n_t = 1000$).

increased Monte-Carlo samples. This figure illustrates the poor estimates we obtain when using small size ($n_t < 50$) of Monte-Carlo samples for validation. On the contrary, the Feuillard's design allows to obtain a good approximation of the true predictivity coefficient even for small sample sizes.

All these observations have to be validated with other tests with different dimensionality, complexity and sample sizes. Future works will perform comparisons of this sequential design with other designs and with cross-validation methods. An industrial nuclear application will also be shown during the conference presentation.

5 CONCLUSIONS

In this paper, we have proposed to look at two special problems when fitting metamodel to small-size data samples in practice. This problem is relevant when one needs to perform uncertainty or sensitivity analysis on cpu time consuming computer code. We have paid attention to the initial input design. Our numerical tests concentrate on the popular Gaussian process metamodel and on the LHS. This type of design, developed thirty years ago, is the most widely used in industrial applications. We have shown that an excellent way to optimize its properties, in the objective of the best metamodel fit, is to use the discrepancy measures, especially the wrap-around L^2 discrepancy. An alternative strategy, if possible, would be to use some adaptative designs. For the Gaussian process metamodel, this kind of design is well-adapted due to the availability of the variance expression (the "error" of the metamodel) (Jourdan 2008).

In a second step, we have looked at the metamodel validation process and have shown that the usual methods can provide biased results. We propose to use a recent algorithm, which puts prediction points in the unfilled zones of the learning sample design. Therefore, a minimal number of points is required to obtain a good estimation of the metamodel predictivity.

REFERENCES

De Rocquigny, E., N. Devictor, and S. Tarantola (Eds.) (2008, to appear). *Uncertainty in industrial practice*. Wiley.

Fang, K.-T., R. Li, and A. Sudjianto (2006). *Design and modeling for computer experiments*. Chapman & Hall/CRC.

Feuillard, V. (2007). *Analyse d'une base de donn´ees pour la calibration d'un code de calcul*. Th'ese de l'Université Pierre et Marie Curie - Paris VI.

Hickernell, F. (1998). A generalized discrepancy and quadrature error bound. *Mathematics of Computation 67*, 299–322.

Jin, R., W. Chen, and A. Sudjianto (2005). An efficient algorithm for constructing optimal design of computer

experiments. *Journal of Statistical Planning and Inference 134*, 268–287.

Jourdan, A. (submitted, 2008). How to repair a second-order surface for computer experiments by kriging? *Available at URL: http://hal.archives-ouvertes.fr/hal-00175059/fr/*.

Kleijnen, J. and R. Sargent (2000). A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research 120*, 14–29.

Liefvendahl, M. and R. Stocki (2006). A study on algorithms for optimization of Latin hypercubes. *Journal of Statistical Planning and Inference 136*, 3231–3247.

Marrel, A., B. Iooss, F. Van Dorpe, and E. Volkova (2008). An efficient methodology for modeling complex computer codes with Gaussian processes. *Computational Statistics and Data Analysis doi:10.1016/j.csda.2008.03.026*.

Matheron, G. (1970). *La théorie des variables r´egionalis´ees et ses applications*. Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau, Fascicule 5. Ecole des Mines de Paris.

O'Hagan, A. (2006). Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering and System Safety 91*, 1290–1300.

Park, J.-S. (1993). Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning and Inference 39*, 95–111.

Sacks, J., W. Welch, T. Mitchell, and H. Wynn (1989). Design and analysis of computer experiments. *Statistical Science 4*, 409–435.

Santner, T., B. Williams, and W. Notz (2003). *The design and analysis of computer experiments*. Springer.

Simpson, T., J. Peplinski, P. Kock, and J. Allen (2001). Metamodel for computer-based engineering designs: survey and recommandations. *Engineering with Computers 17*, 129–150.