ooDACE toolbox

A Matlab Kriging toolbox: Getting started

Ivo Couckuyt Tom Dhaene Piet Demeester

3rd June 2013

Contents

1 Introduction

The **ooDACE** [3, 2] toolbox is a versatile Matlab toolbox that implements the popular Gaussian Process based Kriging surrogate models. Kriging is in particular popular for approximating (and optimizing) deterministic computer experiments [8, 16, 22]. The typical usage of the toolbox is to construct a Kriging model of a dataset obtained by (deterministic) computer simulations or measurements. Afterwards the Kriging surrogate can be fully exploited instead of the (more expensive) simulation code. The toolbox is aimed for solving complex applications (expensive simulation codes, physical experiments, ...) and for researching new Kriging extensions and techniques.

Section 2 discusses the key mathematical formulae of different types of Kriging and gives some insights into various properties. Section 3 demonstrates the usage of the ooDACE toolbox and, additionally, explains the structure of the toolbox.

1.1 Download

See the download page at `http://sumo.intec.ugent.be/?q=ooDACE`.

1.2 Requirements

The ooDACE toolbox takes advantage of the new Object Oriented system (*classdef*) available from Matlab 2008b (v7.7) onwards. By default, the *oodacefit* and *demo* scripts of the ooDACE toolbox use the *fmincon* optimization routine from the Matlab Optimization toolbox. Support for other optimization strategies can easily be used if a wrapper class is coded for it, see Section 3 and Figure 3.2. To that end, full support for the third-party SQPLab optimization package [1] (`http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html`) is included as well as support for the genetic algorithm of the Matlab Global Optimization toolbox.

2 Theory

2.1 Kriging

Kriging (Gaussian Process interpolation) is a surrogate model to approximate deterministic noise-free data, and has proven to be very useful for tasks such as optimization [8], design space exploration, visualization, prototyping, and sensitivity analysis [22]. A thorough mathematically treatment of Kriging is given in [17, 4]. The popularity of Kriging has generated a large body of research, including several extensions to Kriging to handle different problem settings, e.g. by adding gradient information in the prediction [14], or by approximating stochastic simulations [19].

In the remainder of this Section we will give a brief overview of each type of Kriging available in the ooDACE toolbox.

2.1.1 Kriging

Basically, Kriging is a two-step process: first a regression function $f(\mathbf{x})$ is constructed based on the data, and, subsequently, a Gaussian process Z is constructed through the residuals.

$$Y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}), \tag{2.1}$$

where $f(\mathbf{x})$ is a regression (or trend) function and Z is a Gaussian process with mean 0, variance σ^2 and a correlation matrix Ψ.

Depending on the form of the regression function Kriging has been prefixed with different names. *Simple* Kriging assumes the regression function to be a known constant, i.e., $f(\mathbf{x}) = 0$. A more popular version is *ordinary* Kriging, which assumes a constant but unknown regression function $f(\mathbf{x}) = \alpha_0$. Though, other, more complex, trend functions are possible such as linear or quadratic polynomials. In general, *universal* Kriging treats the trend function as a multivariate polynomial, namely,

$$f(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i b_i(\mathbf{x}), \tag{2.2}$$

where $b_i(\mathbf{x})$ are $i = 1 \dots p$ basis functions (e.g., the power base for a polynomial) and $\alpha = (\alpha_1, \dots, \alpha_p)$ denotes the coefficients. The idea is that the regression function captures the largest variance in the data (the general trend) and that the Gaussian Process interpolates the residuals. In fact, the regression function $f(\mathbf{x})$ is actually the mean of the broader Gaussian Process Y. However, selecting the correct regression function is a difficult problem, hence, the regression function is often chosen constant (=ordinary Kriging).

Consider a set of n samples, $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ in d dimensions and associated function values, $\mathbf{y} = \{y^1, \ldots, y^n\}$. Essentially, the regression part is encoded in the $n \times p$ model matrix F,

$$F = \begin{pmatrix} b_1(\mathbf{x^1}) & \cdots & b_p(\mathbf{x^1}) \\ \vdots & \vdots & \vdots \\ b_1(\mathbf{x^n}) & \cdots & b_p(\mathbf{x^n}) \end{pmatrix},$$

while the stochastic process is mostly defined by the $n \times n$ correlation matrix Ψ,

$$\Psi = \begin{pmatrix} \psi(\mathbf{x}^1, \mathbf{x}^1) & \ldots & \psi(\mathbf{x}^1, \mathbf{x}^n) \\ \vdots & \ddots & \vdots \\ \psi(\mathbf{x}^n, \mathbf{x}^1) & \ldots & \psi(\mathbf{x}^n, \mathbf{x}^n) \end{pmatrix},$$

where $\psi(\cdot, \cdot)$ is the correlation function. $\psi(\cdot, \cdot)$ is parametrized by a set of hyperparameters θ, which are identified by Maximum Likelihood Estimation (MLE), see Section 2.3. Subsequently, the prediction mean and prediction variance of Kriging are derived, respectively, as,

$$\mu(\mathbf{x}) = M\alpha + r(\mathbf{x}) \cdot \Psi^{-1} \cdot (\mathbf{y} - F\alpha), \tag{2.3}$$

$$s^2(\mathbf{x}) = \sigma^2 \left(1 - r(\mathbf{x})\Psi^{-1} r(\mathbf{x})^T + \frac{(1 - F^T\Psi^{-1} r(\mathbf{x})^T)}{F'\Psi^{-1}F}\right), \tag{2.4}$$

where $M = \begin{pmatrix} b_1(\mathbf{x}) & b_2(\mathbf{x}) & \ldots & b_p(\mathbf{x}) \end{pmatrix}$ is the model matrix of the predicting point $\mathbf{x}$, $\alpha = (F^T\Psi^{-1}F)^{-1}F^T\Psi^{-1}\mathbf{y}$ is a $p \times 1$ vector denoting the coefficients of the regression function, determined by Generalized Least Squares (GLS), and $r(\mathbf{x}) = \begin{pmatrix} \psi(\mathbf{x}, \mathbf{x}^1) & \ldots & \psi(\mathbf{x}, \mathbf{x}^n) \end{pmatrix}$ is an $1 \times n$ vector of correlations between the point $\mathbf{x}$ and the samples X. The process variance σ^2 is given by $\frac{1}{n}(\mathbf{y} - F\alpha)^T\Psi^{-1}(\mathbf{y} - F\alpha)$.

Note that Kriging, as formulated here, is an interpolation technique. This is easily seen by substituting the i^{th} sample point $\mathbf{x}^i$ in Equation (2.3) and considering that $r(\mathbf{x}^i)$ is the i^{th} column of Ψ, hence, $r(\mathbf{x}^i) \cdot \Psi^{-1}$ is an unit vector e_i with a 1 at the i^{th} position,

$$\mu(\mathbf{x}^i) = M\alpha + e_i \cdot (\mathbf{y} - F\alpha) = M\alpha + y_i - M\alpha = y^i. \tag{2.5}$$

Partial derivatives of the prediction mean with respect to a test point $\mathbf{x}$ are given by,

$$\frac{\partial \mu(\mathbf{x})}{\partial x_i} = \frac{\partial M}{\partial x_i}\alpha + \frac{\partial r(\mathbf{x})}{\partial x_i}\Psi^{-1}(\mathbf{y} - F\alpha) \tag{2.6}$$

where the matrix $\frac{\partial M}{\partial x_i}$ and vector $\frac{\partial r(\mathbf{x})}{\partial x_i}$ are obtained by taking the derivatives of the individual entries respectively, namely,

$$\frac{\partial M}{\partial x_i} = \begin{pmatrix} \frac{\partial b_1(\mathbf{x})}{\partial x_i} & \frac{\partial b_2(\mathbf{x})}{\partial x_i} & \cdots & \frac{\partial b_p(\mathbf{x})}{\partial x_i} \end{pmatrix},$$

$$\frac{\partial r(\mathbf{x})}{\partial x_i} = \begin{pmatrix} \frac{\partial \psi(\mathbf{x}, \mathbf{x}^1)}{\partial x_i} & \cdots & \frac{\partial \psi(\mathbf{x}, \mathbf{x}^n)}{\partial x_i} \end{pmatrix}.$$

2.1.2 Co-Kriging

Co-Kriging (a special case of multi-task or multi-output Gaussian Processes) exploits the correlation between fine and coarse model data to enhance the prediction accuracy. The ooDACE toolbox uses the autoregressive co-Kriging model of Kennedy et al. [12]. Consider two sets of samples, $X_c = \{\mathbf{x}_c^1, \ldots, \mathbf{x}_c^{n_c}\}$ and $X_e\{\mathbf{x}_e^1, \ldots, \mathbf{x}_e^{n_e}\}$, with dimension d obtained from the low-fidelity (cheap) and high-fidelity (expensive) simulator, respectively. The associated function values are denoted by $\mathbf{y}_c = \{y_c^1, \ldots, y_c^{n_c}\}$ and $\mathbf{y}_e = \{y_e^1, \ldots, y_e^{n_e}\}$.

Creating a co-Kriging model can be interpreted as constructing two Kriging models in sequence. First a Kriging model $Y_c(\mathbf{x})$ of the coarse data $(X_c, \mathbf{y}_c)$ is constructed. Subsequently, the second Kriging model $Y_d(\mathbf{x})$ is constructed on the residuals of the fine and coarse data $(X_e, \mathbf{y}_d)$, where $\mathbf{y}_d = \mathbf{y}_e - \rho \cdot \mu_c(X_e)$. The parameter ρ is estimated as part of the MLE of the second Kriging model. Note that the configuration (the choice of the correlation function, regression function, etc.) of both Kriging models can be adjusted separately for the coarse data and the residuals, respectively.

The resulting co-Kriging interpolant is then defined similarly as Equation (2.3),

$$\mu(\mathbf{x}) = M\alpha + r(\mathbf{x}) \cdot \Psi^{-1} \cdot (\mathbf{y} - F\alpha), \tag{2.7}$$

where the block matrices M, F, $r(\mathbf{x})$ and Ψ can be written in function of the two separate Kriging models $\mu_c(\mathbf{x})$ and $\mu_d(\mathbf{x})$:

$$r(\mathbf{x}) = \begin{pmatrix} \rho \cdot \sigma_c^2 \cdot r_c(\mathbf{x}) & \rho^2 \cdot \sigma_c^2 \cdot r_c(\mathbf{x}, X_f) + \sigma_d^2 \cdot r_d(\mathbf{x}) \end{pmatrix},$$

$$\Psi = \begin{pmatrix} \sigma_c^2 \cdot \Psi_c & \rho \cdot \sigma_c^2 \cdot \Psi_c(X_c, X_f) \\ \mathbf{0} & \rho^2 \cdot \sigma_c^2 \cdot \Psi_c(X_f, X_f) + \sigma_d^2 \cdot \Psi_d \end{pmatrix},$$

where $(F_c, \sigma_c, \Psi_c, M_c)$ and $(F_d, \sigma_d, \Psi_d, M_d)$ are matrices obtained from the Kriging models $Y_c(\mathbf{x})$ and $Y_d(\mathbf{x})$, respectively (see Section 2.1.1). In particular, σ_c^2 and σ_d^2 are process variances, while $\Psi_c(\cdot, \cdot)$ and $\Psi_d(\cdot, \cdot)$ denote correlation matrices of two datasets using the optimized $\theta_1 \ldots, \theta_n$ parameters and correlation function of the Kriging models $Y_c(\mathbf{x})$ and $Y_d(\mathbf{x})$, respectively. An illustration of co-Kriging on an one-dimensional example is shown in Figure 2.1.

2.1.3 Blind Kriging

As the actual full behavior of the response is unknown it is often hard to choose which trend function $f(\mathbf{x})$ (the mean of the Gaussian Process) to use for a given problem. Feature selection methods [6] offer the possibility to identify the most plausible interactions occurring in the data. Blind Kriging [11] extends Kriging with a Bayesian feature selection method.

The goal of blind Kriging is to efficiently determine the basis functions b_i (features) that captures the most variance in the sample data. To that end, a set of candidate functions is considered from which to choose from. In the ideal case the sample data is almost fully represented by the chosen trend function and the stochastic process $Z(\mathbf{x})$ has little or no influence.

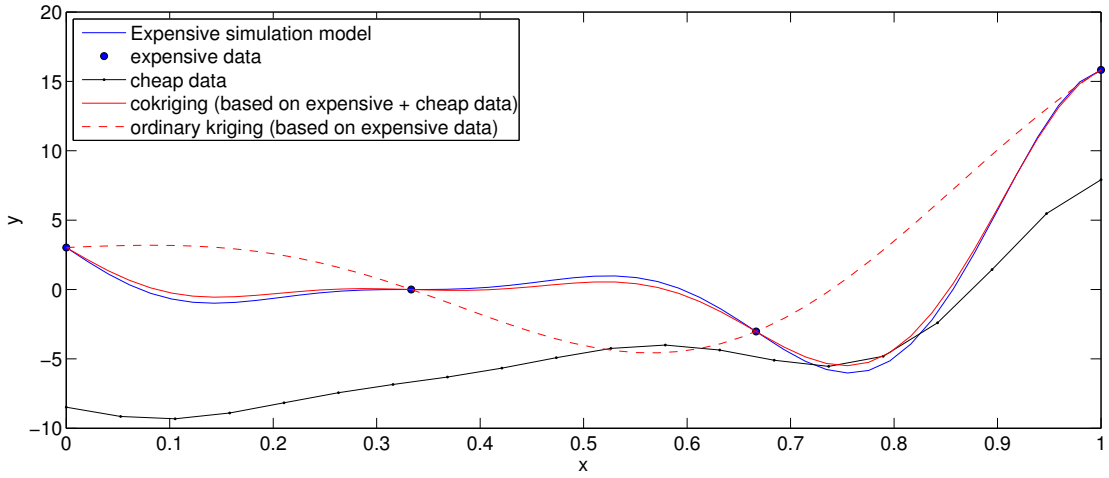


Figure 2.1: Kriging and co-Kriging applied to a 1-dimensional mathematical example. Co-Kriging interpolates the fine data and is further corrected by the coarse data.

Consider an existing Kriging model $Y(\mathbf{x})$, e.g., with a constant regression function (ordinary Kriging). The idea is to select new features to be incorporated in the regression function of this Kriging model, taking into account features that are already part of the regression function of this Kriging model. To that end, the whole set of candidate functions c_i is used to fit the data in a linear model, i.e.,

$$g(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i b_i(\mathbf{x}) + \sum_{i=1}^{t} \beta_i c_i(\mathbf{x}),$$

where t denotes the number of candidate functions.

The first part of this Equation is the regression function of Kriging and, hence, the coefficients α have already been determined independently of $\beta = (\beta_1, \ldots, \beta_t)$. The estimation of β provides a relevance score of the candidate features. A frequentist estimation of β (e.g., least-squares solution) would be a straightforward approach to rank the features. However, this is not always possible as the number of candidate features is often higher than the number of samples available. For instance, considering all possible interactions up to the quadratic effect in four dimensions the number of candidate features is $t = 3^4 = 81$. To that end, a Gaussian Prior distribution is introduced for β,

$$\beta \sim \mathcal{N}(\mathbf{0}, \tau^2 R), \tag{2.8}$$

where $R = U^{-1}\Psi(U^{-1})^T$ and U is the model matrix, namely, a design matrix with t rows. Furthermore, the choice of correlation functions is restricted to the product correlation form,

$$\psi(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \psi_j(|x_j - x'_j|),$$

the variance-covariance matrix R can be constructed independently of the number of dimensions,

$$R_j = U_j^{-1} \psi_j (U_j^{-1})^T,$$

where U_j is the model matrix of the samples for factors $j = 1 \dots d$. Thus the number of considered features can be chosen per dimension and afterwards the full matrix R is obtained by taking the Kronecker product,

$$R = \bigotimes_{j=1}^{d} R_j.$$

While the Bayesian variable selection is able to handle arbitrarily high-order effects, the matrix R grows quite rapidly. Hence, it may be useful to consider the special case where only linear effects, quadratic effects and two-factor interactions are identified. The total set of candidate functions is then defined by $c_i(\mathbf{x})$, where $i = 1 \dots t = 2d^2$. Note that t does not scale exponentially as above, but still the matrix R would already require $(t+1) \times (t+1)$ $(> 4d^4)$ entries.

Let U_j be 3×3 orthogonal polynomial coded [23] matrices, then

$$R_j = U_j^{-1} \psi_j (U_j^{-1})^T = U_j^T \psi_j U_j = \begin{pmatrix} 3 + 4\psi_j(1) + 2\psi_j(2) & 0 & -\sqrt{2}(\psi_j(1) - \psi_j(2)) \\ 0 & 3(1 - \psi_j(2)) & 0 \\ -\sqrt{2}(\psi_j(1) - \psi_j(2)) & 0 & 3 - 4\psi_j(1) + \psi_j(2) \end{pmatrix}, \quad (2.9)$$

While there is some (negative) correlation between mean and quadratic effects (see Equation 2.9), [9, 10] propose to only use the information of the diagonal of R_j. Normalizing to the mean $3 + 4\psi_j(1) + 2\psi(2)$ (first entry of the diagonal of R_j) the variance-covariance matrix R can be expressed as follows. For ease of notation let $\psi(\mathbf{x})$ be a vector of length d,

$$\psi(\mathbf{x}) = \begin{pmatrix} \psi_1(\mathbf{x}) \\ \vdots \\ \psi_d(\mathbf{x}) \end{pmatrix},$$

the vectors $\mathbf{r_l}$ and $\mathbf{r_q}$ of length d are then defined by,

$$\mathbf{r}_l = \frac{3 - 3\psi(2)}{3 + 4\psi(1) + 2\psi(2)}, \quad (2.10)$$

$$\mathbf{r}_q = \frac{3 - 4\psi(1) + \psi(2)}{3 + 4\psi(1) + 2\psi(2)}, \quad (2.11)$$

finally let $\mathbf{l_i}$ be the vector where element $l_{i,j} = 1$ if β_i includes the linear effect of factor j and 0 otherwise. In addition, define $\mathbf{q_i}$ as the vector where element $q_{i,j} = 1$ if β_i includes the quadratic effect of factor j and 0 otherwise. Then the diagonal matrix R is defined as,

$$R = \begin{pmatrix} \mathbf{r}_l^{l_1} \cdot \mathbf{r}_q^{q_1} & 0 & \dots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{r}_l^{l_{t+1}} \cdot \mathbf{r}_q^{q_{t+1}} \end{pmatrix}. \tag{2.12}$$

Note that, as the correlations between mean and quadratic effects have been dropped from Equation (2.9) the matrix R is in fact an estimation of the real correlation matrix.

Having constructed the covariance matrix R by any means explained above, the posterior of β is estimated by,

$$\hat{\beta} = \frac{\tau^2}{\sigma^2} R M_c^T \Psi^{-1} (\mathbf{y} - M \cdot \mathbf{a}), \tag{2.13}$$

$$var(\hat{\beta}) = \tau^2 (R - \frac{\tau^2}{\sigma^2} R M_c^T \Psi^{-1} M_c R), \tag{2.14}$$

where M_c is the $n \times (t+1)$ model matrix of all candidate variables, M is the model matrix of all currently chosen variables and Ψ is the correlation matrix of the samples.

The coefficients $\hat{\beta}$ obtained through this Bayesian variable ranking method quantifies the importance of the associated candidate feature with respect to the data. There are several heuristics proposed to identify the best set of variables to approximate the data. Originally [10], the feature selection consisted of a greedy forward selection procedure, iteratively adding candidate variables with highest standardized coefficients. In blind Kriging [11] the standardized coefficient is replaced with the absolute value of $\hat{\beta}$, delivering similar results while easier to compute. Note that the first term of Equation 2.13 is a constant and does not influence the end results, i.e., $\frac{\tau^2}{\sigma^2}$ is set to 1.

The advantage of choosing this Bayesian variable selection method over other techniques is the tight coupling with Kriging's correlation matrix Ψ, thus taking advantage of already available information. Moreover, this variable selection method incorporates the important variable selection principles - effect hierarchy[1] and effect heredity[2] [7] - in the prior belief of β. In other words, the chosen features should form a simple and interpretable regression function.

In summary, constructing blind Kriging models can be seen as a two stage process. In the first phase an ordinary Kriging model, namely, a Kriging model with a constant regression function, is constructed and θ parameters are estimated. In a second phase the regression function of this initial Kriging model is extended with promising features according to the estimated $\hat{\beta}$ coefficients, generating a series of intermediate Kriging

[1] Effect hierarchy denotes that low order effects (e.g., individual variables) should be chosen before high order effects (e.g., interactions of variables)

[2] Effect heredity states that an effect cannot be important until its parent effect is also important

models. When these intermediate Kriging models stop improving on the leave-one out cross validation prediction error, the search is halted (though a look-ahead of n steps can be used to avoid local optima). The current best set of features is then chosen to construct the final blind Kriging model, re-estimating the θ parameters.

2.1.4 Stochastic Kriging

While the interpolation property (see Equation (2.5)) of Kriging is advantageous for many (deterministic) simulation problems, it might produce undesired results when dealing with stochastic simulations and/or in the presence of noise. Stochastic Kriging [19] extends Kriging for approximation instead of interpolation, also known as regression Kriging. In a way, this type of Kriging most closely resembles standard Gaussian Process regression.

The noise is modeled as a separate Gaussian process $\xi(\mathbf{x})$ with mean 0, and covariance matrix $\sum$,

$$\xi \sim \mathcal{GP}(0, \Sigma).$$

The stochastic Kriging predictor then becomes,

$$\hat{y}(\mathbf{x}) = M\alpha + r(\mathbf{x}) \cdot (\Psi + \frac{1}{\sigma^2}\sum)^{-1} \cdot (\bar{\mathbf{y}} - F\alpha), \tag{2.15}$$

where $\frac{1}{\sigma^2}\sum$ is a matrix resembling noise-to-signal ratios and $\bar{\mathbf{y}}$ is a vector containing the average function values of the repeated simulations for each sample. Note that if the entries of Σ are zero (no noise) the formula is the same as Equation (2.3) and will interpolate the data exactly like universal Kriging.

Depending on the type and distribution of noise the covariance matrix $\sum$ has different forms. In stochastic simulation Σ can be created based on repeated simulations and, in its simplest form, can be defined as,

$$\Sigma = \begin{pmatrix} var(\mathbf{y}^1) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & var(\mathbf{y}^n) \end{pmatrix},$$

where $var(\mathbf{y}^i)$ is the variance between the repeated simulations of data point i.

However, stochastic Kriging (or regression Kriging) is also useful for deterministic (but noisy) simulation problems (or measurements), where repeated simulations are not available. Assuming the noise is homogeneous distributed across the input space the matrix Σ consists of a scalar value (10^λ) on its diagonal, i.e., $\sum = 10^\lambda I_n$, where I_n is the $n \times n$ identity matrix. The variable λ (amount of noise) is estimated as part of the likelihood optimization of Kriging. Logically, in case of heterogeneous noise the matrix Σ has different values on the diagonal, which introduces more variables in the likelihood optimization problem.

2.1.5 Miscellaneous

In addition to the distinct types of Kriging models the ooDACE toolbox also incorporates several smaller extensions and useful tools. A number of those tools are described in this Section.

2.1.5.1 Re-interpolation of the prediction variance

When using stochastic Kriging (or regression Kriging) the prediction variance is not zero anymore in the sample points. However, this is actually a desired property for many algorithms to achieve some form of space-fillingness (e.g., maximizing the expected improvement or prediction variance). To that end, Forrester et al. [5] suggest a re-interpolation of the prediction variance. In essence, this is done by ignoring the Σ matrix in the respected formulae. In particular, the process variance σ^2 and the prediction variance formulations are taken from standard Kriging, see Equation (2.4), which differ only from stochastic Kriging with respect to the covariance matrix Σ.

2.1.5.2 Leave-one-out cross validation

The leave-one-out cross validation (or cross-validated prediction error; cvpe) score [21, 11] can be efficiently calculated as follows:

$$H = F(F^T F)^{-1} F^T,$$

$$\mathbf{d} = \mathbf{y} - F\alpha,$$

$$cvpe = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{(\Psi^{-1})_{i,:} \cdot \left(\mathbf{d} + H_{:,i} \cdot \frac{\mathbf{d}_i}{1 - H_{ii}} \right)}{(\Psi^{-1})_{ii}} \right)^2, \tag{2.16}$$

where $A_{i,:}$, $A_{:,i}$ denote the i^{th} row and column, respectively, for a matrix A while A_{ii} is the i^{th} entry on the diagonal.

2.1.5.3 Integrated mean square error

The integrated mean square error (imse) [16] is another criterion to measure the accuracy of a Kriging model and is defined as,

$$imse = \int_A s^2(\mathbf{x}) dA, \tag{2.17}$$

where A denotes the input domain. Naturally, it is impossible to evaluate this integral analytically, hence, approximation methods should be used such as trapezoidal numerical integration or Monte carlo sampling.

2.1.5.4 Error on a test set

A safe way to measure the accuracy of a surrogate model is to use an independent test set $X_{test} = \{\mathbf{x}^1_{test}, \ldots, \mathbf{x}^{n_{test}}_{test}\}$ with associated function values $y^1_{test}, \ldots, y^{n_{test}}_{test}$ (if available).

$$error = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left(\mu(\mathbf{x}^i_{test}) - y^i_{test}\right)^2. \tag{2.18}$$

2.1.5.5 Robustness criterion

Siem et al. [18] propose the robustness-criterion (rc) to measure the stability of an ordinary Kriging model with respect to simulation errors. Let $\gamma = \Psi^{-1} \cdot (\mathbf{y} - F\alpha)$ then the absolute and relative robustness criterion are defined as,

$$rc_{absolute} = |\gamma|^2_2, \tag{2.19}$$

$$rc_{relative} = {}^{|\gamma|^2_2}/_{|\gamma|_\infty}. \tag{2.20}$$

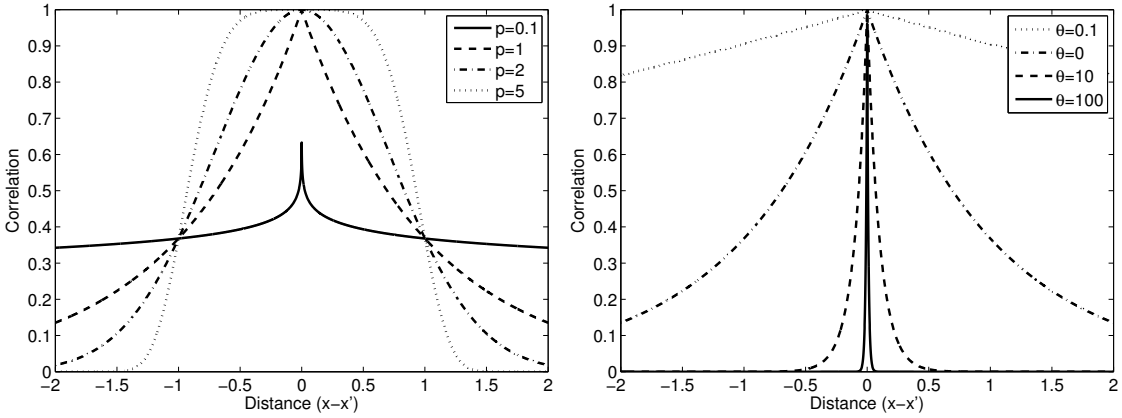
2.2 Correlation functions

Arguably, the choice of correlation function is crucial to create an accurate Kriging surrogate model, whether it is universal Kriging, co-Kriging, stochastic Kriging, etc. A popular class of *stationary* correlation functions is defined by,

$$\psi(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{i=1}^{d} \theta_i |x_i - x'_i|^p\right).$$

Note that these correlation functions only depend on the distance between the two points $\mathbf{x}$ and $\mathbf{x}'$. The smaller the distance between two points, the higher the correlation and, hence, the more the prediction of one point is influenced by the other, i.e., their function values are closer together. Similarly, if the distance increases the correlation drops to zero. The rate and manner at which this happens is governed by several parameters. The parameter p determines the initial drop in correlation as distance increases, see Figure 2.2a. Often p is set to two (=the Gaussian correlation function) which assumes that the data represents a smooth, continuous surface. A lower value of p, e.g., $p = 1$, is more suitable for a more rough (sharp/erratic) response as it permits a more substantial difference in function values for points close together.

The second set of parameters, $\theta_1, \ldots, \theta_d$, describes the influence sphere of a point on nearby points for each dimension, i.e., how fast the correlation drops to zero, see 2.2b. Usually, p is set fixed while the parameters $\theta_1, \ldots, \theta_d$ are identified using Maximum Likelihood Estimation (MLE). The MLE's of $\theta_1, \ldots, \theta_d$ are useful as they describe the amount of variation in each dimension i. A high value of θ_i means points have less influence on each other and, thus, similar points in the input space can have a very different response value (highly non-linear behavior in dimension i). On the other hand,



(a) (b)

Figure 2.2: Examples of one-dimensional correlation functions: a) with varying parameter p for $\theta = 1$) b) with varying parameter θ for $p = 1$.

a low value of θ_i indicates that a point is correlated with points that are farther away (a more linear behavior).

The ooDACE toolbox offers three instances of this well-known class of correlation functions,

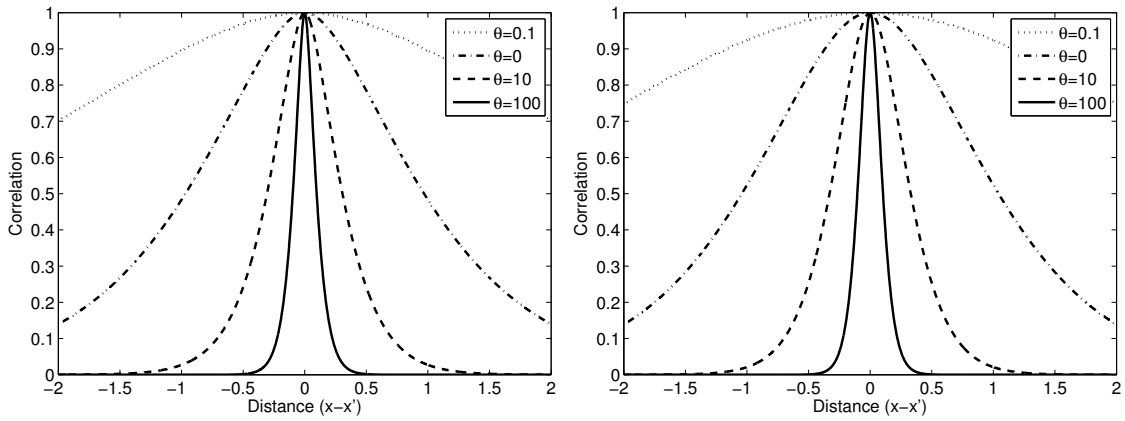
- The Gaussian correlation function ($p = 2$, *corrgauss*)
- The exponential correlation function ($p = 1$, *correxp*)
- A generic version where p is included in the likelihood optimization (*corrgaussp*)

While, arguably, these are the most frequently used correlation functions - and in particular the Gaussian correlation function - to solve engineering problems, the Matérn class of correlation functions [20] might be actually more useful. The ooDACE toolbox implements two instances of the Matérn correlation function, for $\nu = 3/2$ (*corrmatern32*) and $\nu = 5/2$ (*corrmatern52*), namely,

$$\psi(\mathbf{x}, \mathbf{x}')_{\nu=3/2} = \left(1 + \sqrt{3}l\right) \exp\left(-\sqrt{3}l\right),$$

$$\psi(\mathbf{x}, \mathbf{x}')_{\nu=5/2} = \left(1 + \sqrt{5}l + \frac{5l^2}{3}\right) \exp\left(-\sqrt{5}l\right),$$

with $l = \sqrt{\sum_{i=1}^{d} \theta_i (x_i - x_i')^2}$. The parameter ν of the Matérn correlation functions has a similar role as the p parameter, see Figure 2.3, a higher value is better suited for a rough behavior of the expensive function and vice versa.



(a) (b)

Figure 2.3: The one-dimensional Matérn correlation functions with varying parameter θ: a) for $\nu = 3/2$ b) for $\nu = 5/2$.

2.3 Maximum Likelihood Estimation (MLE)

After choosing the general form of the correlation function for the problem at hand, its hyperparameters are identified using Maximum Likelihood Estimation (MLE). There are several variants of the likelihood that one can optimize, with the marginal likelihood the most widely used.

2.3.1 Marginal likelihood

The natural log of the marginal likelihood is given by,

$$\ln(\mathcal{L}_{marginal}) = \frac{n}{2}\ln(2\pi) + \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\ln(|\Psi|) + \frac{1}{2\sigma^2}(\mathbf{y} - F\alpha)^T\Psi^{-1}(\mathbf{y} - F\alpha).$$

This can be simplified by taking the derivatives with respect to the α and σ^2, equaling it to zero and solving for α and σ^2. When we also remove constant terms the (negative) concentrated ln-likelihood is obtained as used in the ooDACE toolbox,

$$-\ln(\mathcal{L}_{marginal}) = -\frac{n}{2}\ln\sigma^2 - \frac{1}{2}\ln(|\Psi|) \tag{2.21}$$

where $\alpha = (F^T\Psi^{-1}F)^{-1}F^T\Psi^{-1}\mathbf{y}$ and $\sigma^2 = \frac{1}{n}(\mathbf{y} - F\alpha)^T\Psi^{-1}(\mathbf{y} - F\alpha)$, see also Section 2.1.1.

In essence, the first term, $-\frac{n}{2}\ln\sigma^2$, denotes the quality of the fit while the second term, $-\frac{1}{2}\ln(|\Psi|)$, represents a complexity penalty. Thus, the marginal likelihood automatically balances flexibility versus accuracy. However, the marginal likelihood depends on correct specification of the Kriging model for the data (e.g., choice of correlation function) and may not be robust enough when the Kriging model is misspecified.

2.3.2 Pseudo likelihood

Leave-one-out cross-validation is a popular method to assess the accuracy of a surrogate model, especially for interpolation based methods, and, hence, can also be used to tune the hyperparameters of Kriging. The Leave-One-Out (LOO) predictive ln-probability [21, 15] is given by,

$$\ln(\mathcal{L}_{LOO}) = \sum_{i=1}^{n} -\frac{1}{2}\ln(\sigma_i^2) - \frac{(y^i - F\alpha - \mu^i)^2}{2\sigma_i^2} - \frac{1}{2}\ln(2\pi) \tag{2.22}$$

where $\mu^i = y^i - F\alpha - {}^{(\Psi^{-1}\mathbf{y})_i}/_{\Psi_{ii}^{-1}}$ and $\sigma_i^2 = {}^{1}/_{\Psi_{ii}^{-1}}$. In contrast to the marginal likelihood, the LOO predictive ln-probability is independent of the model specification and, thus, may be more robust, though this has not been empirically validated.

3 Practical

3.1 Getting started

Before the ooDACE toolbox can be used you have to include it in Matlab's search path. You can do this manually by running startup, or, if Matlab is started in the root toolbox directory, then startup will be run automatically.

```
startup
```

Now the toolbox is ready to be used. The ooDACE toolbox is designed in an Object Oriented (OO) fashion. It is strongly recommended to exploit the OO design directly, i.e., use the Kriging and Optimizer Matlab classes, see Figures 3.1 and 3.2. Most functionality is implemented in the base class *BasicGaussianProcess*, the derived *Kriging* class differs only in the fact that it automatically normalizes your dataset before fitting. For more information on the classes and their methods please refer to the Doxygen documentation and the source files.

Lets define n as the number of observations and d as the number of input parameters. Then the $n \times d$ input sample matrix is denoted by **samples** (each row is one observation) and the corresponding output values are stored in the $n \times 1$ matrix **values**. The ooDACE toolbox provides a script, *oodacefit.m*, that just takes your dataset (a **samples** and **values** matrix) and, optionally, an options structure and returns a fitted Kriging object, all other parameters are set to some sensible defaults (the options structure is merged with the defaults). The internal call hierarchy of fitting a Kriging object, and predicting some points, is shown by a sequence diagram in Figure 3.3, any part of the fitting (or prediction) process can be easily modified by inheriting from the appropriate class and overriding the desired methods.

The computational complexity of the implementation is governed by the initial fitting and hyperparameter estimation (the prediction can be calculated in linear time). The fitting of a Kriging model involves calculating the likelihood function multiple times to identify a set of hyperparameters. The efficient calculation of the likelihood involves the factorization of the correlation matrix Ψ. Using Cholesky decomposition this requires a time complexity of the order $O(n^3)$ (memory cost is $O(n^2)$), where n is the number of rows/columns (equal to the number of samples for most Kriging variants, except co-Kriging). The Cholesky decomposition is the most expensive part of the code and, thus, to reduce the number of likelihood evaluations, derivative information is utilized by default. The calculation of the likelihood derivatives comes at an extra memory cost (as well as computational cost $O(d)$) which is roughly equal to $O(d \times n^2)$. For low-dimensional problems ($d < 20$) this is not really a problem, however for higher-dimensional problems

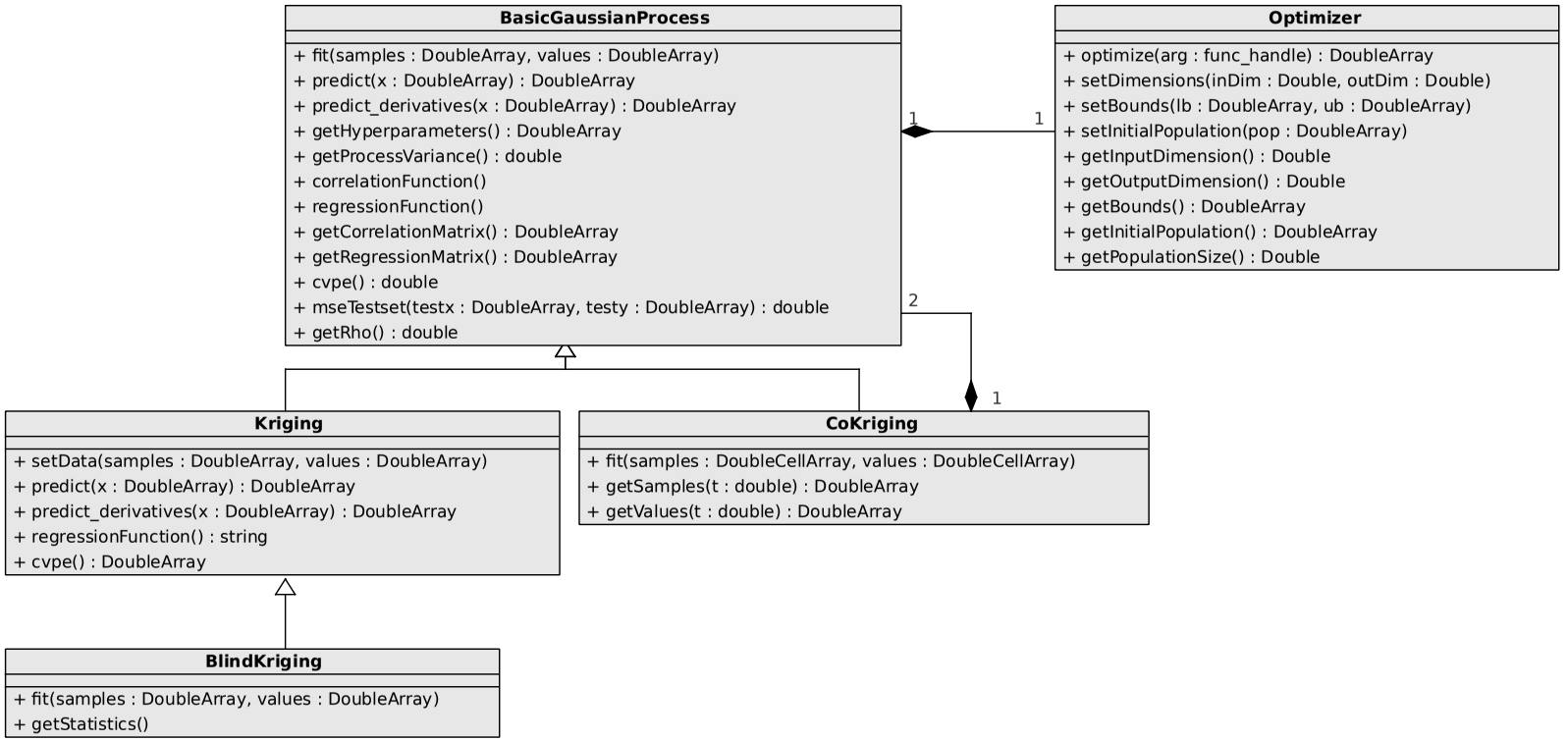


Figure 3.1: Class diagram of the ooDACE toolbox.

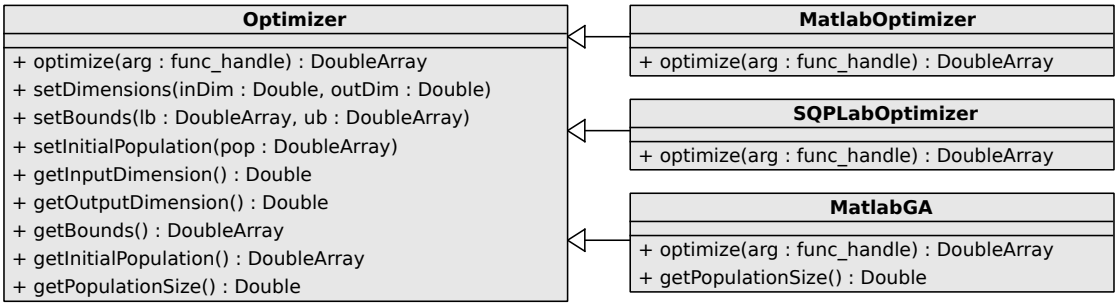


Figure 3.2: Class diagram of the Optimizer class hierarchy.

it may be required use a different optimization strategy which does not use derivative information.

The remainder of this Section presents pseudo-code how to use the ooDACE toolbox for different use cases and types of Kriging models. See the included *demo.m* and *oodacefit.m* scripts for more example code on how to use the ooDACE toolbox, including more advanced features such as using blind Kriging (see the *BlindKriging* class) or how to use regression instead of interpolation. For convenience, wrapper scripts (*dacefit.m*, *predictor.m*) are provided that emulate the DACE toolbox interface (see Section 3.4 for more information).

3.1.1 Kriging

Creating and exploiting a Kriging model requires only two lines of code:

```
k = oodacefit( samples, values );
y = k.predict(x);
```

For more flexibility, e.g., choosing the correlation and regression functions, the user can utilize the *Kriging* classes directly. **lb** and **ub** are $1 \times d$ arrays defining the lower bounds and upper bounds, respectively, needed to optimize the **hyperparameters**. In addition, a set of starting values has to be specified, namely, **hyperparameters0** is also an $1 \times d$ array. Example code to create an universal Kriging model follows:

```
...
% Generate Kriging options structure
opts = Kriging.getDefaultOptions();
opts.hpBounds = [lb ; ub]; % hyperparameter optimization bounds
% configure the SQPLab optimization algorithm (included)
opts.hpOptimizer = SQPLabOptimizer( inDim, 1 );
% create and fit the Kriging model
k = Kriging( opts, hyperparameters0, 'regpoly2', @corrgauss );
k = k.fit( samples, values );
k = k.cleanup(); % optional: only needed for very large datasets
```

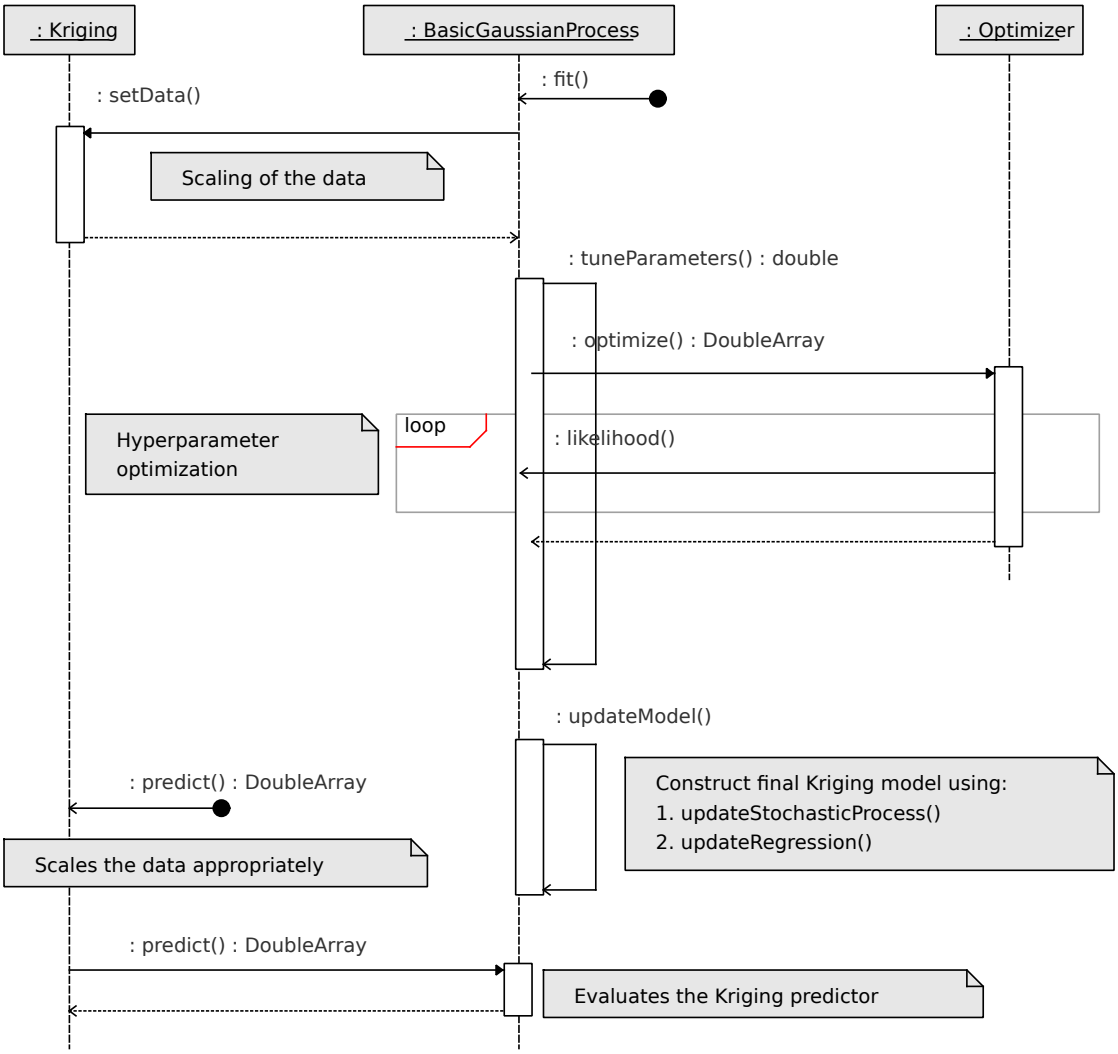


Figure 3.3: Sequence diagram of constructing a Kriging class.

```
% k represents the approximation and can now be used, e.g.,
[y mse] = k.predict( [1 2] )
...
```

The optional call to *cleanup()* after fitting a Kriging model clears some temporary, unused variables from memory to reduce memory usage, this may be especially useful for large datasets. Once a Kriging model has been constructed subsequent calls to the fit method of Kriging will use the previously optimized hyperparameters on the new data. This is useful for, e.g., calculating a 20-fold cross-validation score for which the hyperparameters need to remain fixed, some Matlab pseudo-code follows,

```
...
for i=1:20
  % fit cross-validated Kriging model
  k_xval = k.fit( samples(fold{i},:), values(fold{i},:) );
  % calculate error for this fold
  xval(i,:) = mean( (k_xval.predict( samples(fold{i},:) ) - ...
                     values(fold{i},:)).^2 );
end
% final score is the mean of all fold errors
xvalScore = mean( xval );
```

Here *fold{i}* are indices to a subset of the dataset for fold i. Note that leave-one-out crossvalidation (using the mean square error function) can be obtained directly using *k.cvpe()*, see Section 2.1.5.2.

3.1.2 Co-Kriging

The *oodace* script automatically creates a co-Kriging model if **samples** and **values** are cell arrays of length two. The first elements of **samples** and **values** describe the cheap data, represented by a $n_c \times d$ matrix (*samples{1}*) and a $n_c \times 1$ matrix (*values{1}*). Similarly, the second entries of both cell arrays contain the expensive data. A co-Kriging model is then created by executing:

```
k = oodacefit( samples, values );
y = k.predict(x);
```

On the other hand, the exact optimization strategies to use and other options can be defined when constructing the *CoKriging* class directly,

```
...
% Generate CoKriging options structure
opts = CoKriging.getDefaultOptions();
opts.hpBounds = [lb ; ub]; % hyperparameter optimization bounds
%% configure the optimization algorithms for
```

```
% for cheap data
opts.hpOptimizer{1} = SQPLabOptimizer( dim, 1 );
% for expensive data
optimopts.DerivativeCheck = 'off';
optimopts.Diagnostics = 'off';
optimopts.Algorithm = 'active-set';
optimopts.MaxFunEvals = 1000000;
optimopts.MaxIter = 500;
optimopts.GradObj = 'off';
opts.hpOptimizer{2} = MatlabOptimizer( inDim, 1, optimopts );
%% create and fit the CoKriging model
k = CoKriging( opts, hyperparameters0, 'regpoly0', @correxp );
k = k.fit( samples, values );
% k represents the approximation and can now be used, e.g.,
[y mse] = k.predict( [1 2] )
...
```

The co-Kriging model can be efficiently updated with new expensive data, which involves a re-estimation of the hyperparameters of one of the underlying Kriging models of co-Kriging. This is in contrast to Kriging were subsequent calls to *fit()* keep the hyperparameters fixed. When new expensive data arrives it suffices to update the second entry of the cell arrays samples and values with the new data,

```
...
samples{2} = [samples{2}; samples_new];
values{2} = [values{2}; values_new];
k = k.fit( samples, values );
```

Note that the co-Kriging class does not scale the data due to this feature as that might produce undesired results. It is suggested to scale (or normalize) your data to, e.g., $[0, 1]$, manually before calling the fit method.

3.1.3 Blind Kriging

A blind Kriging model is created using:

```
opts.type = 'BlindKriging';
k = oodacefit( samples, values, opts );
y = k.predict(x);
```

Additional options are,

```
% retune parameters after every iteration
opts.retuneParameters = false;
% maximum order of candidate features to consider (quadratic)
opts.regressionMaxOrder = 2;
```

Similarly to the previous Sections, a blind Kriging model can also be constructed directly by calling the constructor and fit method of the *BlindKriging* class.

3.1.4 Stochastic Kriging

For stochastic simulation problems a stochastic Kriging can be fitted using the *BasicGaussianProcess* class with the following additional options,

```
% Sigma is the intrinsic covariance matirx (=variance of output values)
opts.Sigma = var(values,2);
values = mean(values,2);
% the process variance sigma2 needs to be included in the MLE
opts.sigma20 = Inf; % optional, guess the initial value
opts.sigma2Bounds = [-2 ; 4]; % log10 scale
opts.generateHyperparameters0 = true; % optional, guess the initial value
% explicitly ask for BasicGaussianProcess (=Kriging without scaling)
opts.type = 'BasicGaussianProcess';
k = oodacefit( samples, values, opts );
[y s2] = k.predict(x);
```

On the other hand, when dealing with noisy simulators that are actually deterministic, one can use regression Kriging which tries to identify the amount of noise automatically by including an extra parameter λ in the likelihood optimization. A regression Kriging model is constructed using the *Kriging* class as follows:

```
% regression Kriging
opts.lambda0 = 0;
opts.lambdaBounds = [-5 ; 5]; % log scale
k = oodacefit( samples, values, opts );
[y s2] = k.predict(x);
```

As regression Kriging approximates the data, in contrast to interpolation, the prediction variance will not be zero at the samples. By specifying the option,

```
opts.reinterpolation = true;
```

before fitting the regression Kriging model, *predict()* will return the re-interpolated prediction variance as its second output argument.

3.2 Running the problems provided with ooDACE (demo.m)

The *demo.m* script includes several test cases trying to cover the most important aspects of the ooDACE toolbox. To solve a problem just run the demo script and make your selection of the several test cases, or, you can execute a test case directly by calling, e.g.,

```
demo(4)
```

Each test case creates a landscape plot of the Kriging model, as well as two contour plots of the prediction and the prediction variance (and the derivatives), respectively. These plots are found in Figure 3.4. A quick discussion of the test cases follows.

3.2.1 demo(1) - fitting a standard Kriging model

This test case fits a standard ordinary Kriging model on the Branin function. This is the most simple use of the ooDACE toolbox and is probably the setup most users want to utilize. The output of Matlab is,

```
>> demo(1)

Building model...done

Evaluating model at (2.500000,7.500000).
Prediction mean = 33.294889. Prediction variance = 201.580375.
Derivatives of: prediction mean = (6.158208,11.097571).
   prediction variance = (0.021838,-0.000497).
Leave-one-out crossvalidation: 1029.848737 (using the mean
   squared error function).
Integrated Mean Square Error: 57324.193566.
Marginal likelihood (-log): -3.940479.
Pseudo likelihood (-log): 9.981225.
Process variance: 13110.266468
Sigma(1,1): 0.000000 (first element of intrinsic covariance
   matrix).
Formatted regression function: 0
Calculating derivatives for contour plot... (may take a while).

ans =

Kriging model with correlation function corrmatern32 ( -0.51
   -0.88 )
Average Sigma 5.773e-15
```

Beside the prediction and prediction variance (and their derivatives), several accuracy metrics are available giving an indication of the accuracy of the Kriging model.

3.2.2 demo(2) - fitting a regression Kriging model

This test cases demonstrate how to create a Kriging model for noisy data, we use the Bird function modified by adding some noise. We use the pseudo-likelihood to optimize the hyperparameters instead of the standard marginal likelihood and enable the re-interpolation of the prediction variance. This last option makes sure the prediction

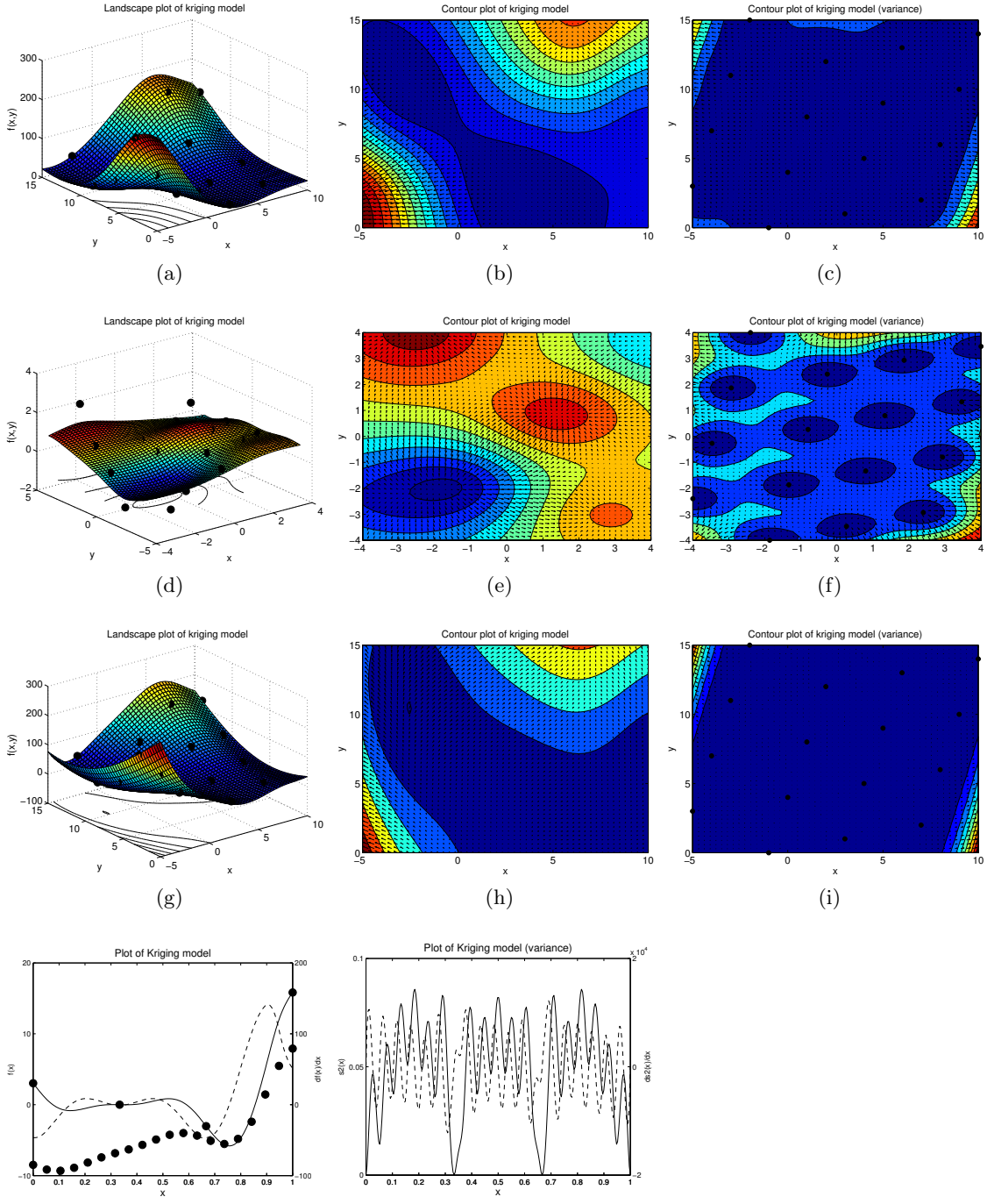
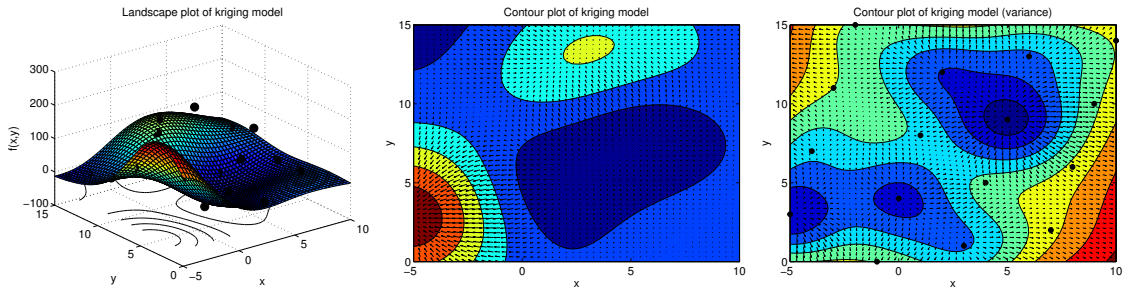

Landscape plot of kriging model
Contour plot of kriging model
Contour plot of kriging model (variance)
(a)
(b)
(c)
Landscape plot of kriging model
Contour plot of kriging model
Contour plot of kriging model (variance)
(d)
(e)
(f)
Landscape plot of kriging model
Contour plot of kriging model
Contour plot of kriging model (variance)
(g)
(h)
(i)
Plot of Kriging model
Plot of Kriging model (variance)
(j)
(k)

Figure 3.4



(l) (m) (n)

Figure 3.4: Plots generated by the several test cases of *demo.m*. The first row (a, b, c) is of the first test case (demo(1)), The second row (d, e, f) of the second test case (demo(2)), etc. The black dots are the samples and the black arrows represents the derivatives.

variance is zero at the samples, which is sometimes desired (e.g., for optimization). In addition, debug mode is enabled and, hence, a debug contour plot of the likelihood surface is calculated, see Figure 3.5.

The output looks like,

```
>> demo(2)

Building model...
... (removed the output of creating the likelihood plot)
done

Evaluating model at (0.000000,0.000000).
Prediction mean = 0.607674. Prediction variance = 0.061207.
Derivatives of: prediction mean = (0.288204,0.442158).
    prediction variance = (0.019058,-0.027821).
Leave-one-out crossvalidation: 0.891207 (using the mean squared
    error function).
Integrated Mean Square Error: 5.072019.
Marginal likelihood (-log): -0.869552.
Pseudo likelihood (-log): 21.184450.
Process variance: 0.526587
Sigma(1,1): 1.000000 (first element of intrinsic covariance
    matrix).
Formatted regression function: 0
Calculating derivatives for contour plot... (may take a while).

ans =
```

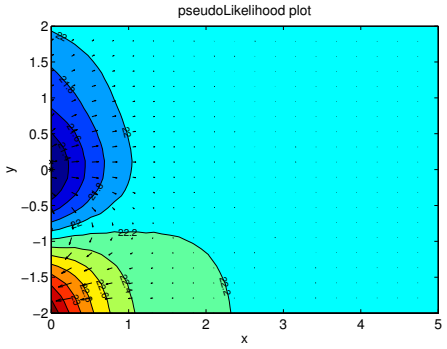


Figure 3.5: Contour plot of the marginal likelihood function for demo(2). The green cross are the optimal values found by the Maximum Likelihood Estimation (MLE), while the green star are the minimum values identified while generating this contour plot. For this particular setup x and y represent the hyperparameters σ^2 and θ_1, respectively. θ_2 is set fixed to the optimal value found by the MLE.

```
Kriging model with correlation function corrmatern32 ( 0.00 0.30
    )
Average Sigma 1.000e+00
```

Note that Sigma(1,1) (= lambda hyperparameter) is larger than zero as we are doing regression instead of interpolation now.

3.2.3 demo(3) - fitting a blind Kriging model

This test case is the same as demo(1), except we are fitting a blind Kriging model. This type of Kriging tries to automatically determine the right regression (trend) function of the data. The output is,

```
>> demo(3)

Building model... done

Evaluating model at (2.500000,7.500000).
Prediction mean = 30.862115. Prediction variance = 21.030948.
Derivatives of: prediction mean = (16.136297,20.972668).
   prediction variance = (0.005540,0.003623).
Leave-one-out crossvalidation: 10.319308 (using the mean squared
    error function).
Integrated Mean Square Error: 30504.350596.
Marginal likelihood (-log): -9.518351.
Pseudo likelihood (-log): -18.507680.
Process variance: 27280.976795
```

```
Sigma(1,1): 0.000000 (first element of intrinsic covariance
    matrix).
Formatted regression function: 1+x1^2+x1
Calculating derivatives for contour plot... (may take a while).

ans =
Kriging model with correlation function corrgauss ( -0.23 -1.22
    )
Average Sigma 5.773e-15
```

As we can see there is a dramatic improvement on the leave-one-out crossvalidation score in comparison to the first test case. The final regression function, without the coefficients, is shown after 'Formatted regression function', namely, $1 + x_1^2 + x_1$.

3.2.4 demo(4) - fitting a co-Kriging model

This test case deals with multi-fidelity data, namely, two datasets modeling the same problem but coming from two different simulators with varying accuracy. Typical we have a dataset from an expensive and a cheap simulator. This data can be combined to enhance accuracy by creating a co-Kriging model. For demonstration purposes two mathematical functions are used here to represent the two simulators. The output looks like,

```
>> demo(4)

Building model...done

Evaluating model at (0.5).
Prediction mean = 0.775430. Prediction variance = 0.082380.
Derivatives of: prediction mean = (5.3669). prediction variance
    = (-1.0102e-08).
Leave-one-out crossvalidation: 0.032823 (using the mean squared
    error function).
Integrated Mean Square Error: 0.050392.
Marginal likelihood (-log): 98.961100.
Pseudo likelihood (-log): 92.816498.
Process variance: 3821.782989
Sigma(1,1): 0.000000 (first element of intrinsic covariance
    matrix).
Formatted regression function: Not available
Rho: 1.931520
Calculating derivatives for plot... (may take a while).

ans =
```

```
Kriging model with correlation function corrmatern32 ( 0.27
    -1.99 )
Average Sigma 2.356e-12
```

3.2.5 demo(5) - fitting a stochastic Kriging model

Finally, this test case demonstrates the stochastic Kriging model. This approximation model is used when dealing with data from stochastic simulations. Often the stochastic simulator provides error bounds on the output noise and/or multiple simulation runs are done to get an estimate of the amount of noise. Stochastic Kriging can use this extra information to improve accuracy. Here we use data from the Branin function with some random noise added to it. The output is,

```
>> demo(5)

Building model...done

Evaluating model at (2.5 7.5).
Prediction mean = -29.394889. Prediction variance =
    11532.981400.
Derivatives of: prediction mean = (-0.000851706   0.00141918).
    prediction variance = (-0.0732731 -0.0511818).
Leave-one-out crossvalidation: 7709.117897 (using the mean
    squared error function).
Integrated Mean Square Error: 2594920.816745.
Marginal likelihood (-log): 99.724451.
Pseudo likelihood (-log): 169.516833.
Process variance: 11532.981459
Sigma(1,1): 1337.630131 (first element of intrinsic covariance
    matrix).
Formatted regression function: 0 Calculating derivatives for
    contour plot... (may take a while).

ans =
Kriging model with correlation function corrgauss ( -1.30 -1.35
    )
Average Sigma 1.762e+04
```

The difference between stochastic Kriging and regression Kriging is that here the matrix Sigma is not included in the maximum likelihood estimation, but is defined a priori by the user (often by replicating the stochastic simulations a number of times).

3.3 Regression tests

The ooDACE toolbox also includes a regression test suite which can be run as follows:

```
runRegressionTests
```

Results of each test are compared against previous saved results (in regressionTests/) and when there are no problems found the output should be,

```
Running test 1...OK.
Running test 2...OK.
Running test 3...OK.
Running test 4...OK.
Running test 5...OK.
```

Small changes between Matlab versions (e.g., *dblquad* versus *integral2* for the *imse* method) and the Optimization toolbox (*fmincon*) will are handled by introducing a tolerance value (set by default to 10^{-6} using the combined relative error) in the comparison tests. If the difference is higher than the tolerance value then the corresponding test will fail and the maximum combined relative error will be displayed in the output. Regression tests are most useful for developers: Before making changes to the code the output of the regression tests (for a particular Matlab setup) can be saved using *runRegressionTests([1:5], 'regressionTests_custom', true);*. After introducing new functionality to the ooDACE toolbox running *runRegressionTests(1:5, 'regressionTests_custom')* again will find any regressions in the code.

3.4 DACE toolbox interface

The ooDACE toolbox provides two scripts *dacefit.m* and *predictor.m* that emulate the behavior of the DACE toolbox [13]. Note, that full compatibility is not provided. The scripts merely aim to ease the transition from the DACE toolbox to the ooDACE toolbox. Example code,

```
krige = dacefit(samples, values, 'regpoly0', 'corrgauss', hyperparameters0, lb, ub )
y = predictor([1 2], krige)
```

Obviously, a lot less code is used to copy the setups described in the previous sections. However, less code means less flexibility (e.g., blind Kriging and regression Kriging are not available using the wrapper scripts). Hence, it is suggested to learn the object oriented interface of ooDACE and use it instead.

4 Contribute

Suggestions on how to improve the ooDACE toolbox are always welcome. For more information please see the feedback page at `http://sumowiki.intec.ugent.be/index.php/Feedback`.

Bibliography

[1] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[2] I. Couckuyt, K. Crombecq, D. Gorissen, and T. Dhaene. Automated response surface model generation with sequential design. In *First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC), Funchal, Portugal*, 2009.

[3] I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of RF and Microwave Computer-Aided Engineering: Advances in design optimization of microwave/rf circuits and systems*, 20(5):492–501, 2010.

[4] A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

[5] A.I.J. Forrester, A.J. Keane, and N.W. Bressloff. Design and analysis of "noisy" computer experiments. *AIAA Journal*, 44(10):2331–2336, 2006.

[6] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003.

[7] M. Hamada and C. F. J. Wu. Analysis of designed experiments with complex aliasing. *Quality Technology*, 24(3):130–137, 1992.

[8] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998.

[9] V. R. Joseph. A bayesian approach to the design and analysis of fractionated experiments. *Technometrics*, 48(2):221–229, 2006.

[10] V. R. Joseph and J. D. Delaney. Functionally induced priors for the analysis of experiments. *Technometrics*, 49:1–11, 2007.

[11] V. R. Joseph, Y. Hung, and A. Sudjianto. Blind kriging: A new method for developing metamodels. *ASME Journal of Mechanical Design*, 130(3):031102–1–8, 2008.

[12] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87:1–13, 2000.

[13] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

[14] M.D. Morris, T.J. Mitchell, and D. Ylvisaker. Design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[16] J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–435, 1989.

[17] T.J. Santner, B.J. Williams, and W.I. Notz. *The design and analysis of computer experiments*. Springer series in statistics. Springer-Verlag, New York, 2003.

[18] A.Y.D. Siem and D. den Hertog. Kriging models that are robust w.r.t. simulation errors. Technical report, Tilburg University, 2006.

[19] J. Staum. Better simulation metamodeling: The why, what, and how of stochastic kriging. In *Proceedings of the Winter Simulation Conference*, 2009.

[20] M.L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, 1999.

[21] S. Sundararajan and S. Sathiya Keerthi. Predictive approach for choosing hyperparameters in gaussian processes. *Neural Computation*, 13:1103–1118, 2001.

[22] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[23] J.C.F. Wu and M. Hamada. *Experiments: Planning, Analysis, and Parameter Design Optimization*. Wiley, New York, 2000.