IMM
INFORMATICS AND MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Kongens Lyngby – Denmark

J. No. DACE1
1.8.2002
HBN/ms

DACE – A MATLAB KRIGING TOOLBOX

VERSION 2.0

Søren N. Lophaven
Hans Bruun Nielsen
Jacob Søndergaard

TECHNICAL REPORT
IMM-REP-2002-12

IMM

DACE
A Matlab Kriging Toolbox
Version 2.0

Søren N. Lophaven
Hans Bruun Nielsen
Jacob Søndergaard

Contents

1. Introduction

This report describes the background for and use of the software package DACE (Design and Analysis of Computer Experiments), which is a Matlab toolbox for working with kriging approximations to computer models.

The typical use of this software is to construct a kriging approximation model based on data from a computer experiment, and to use this approximation model as a surrogate for the computer model. Here, a computer experiment is a collection of pairs of input and responses from runs of a computer model. Both the input and the response from the computer model are likely to be high dimensional.

The computer models we address are deterministic, and thus a response from a model lacks random error, i.e., repeated runs for the same input parameters gives the same response from the model.

Often the approximation models are needed as a part of a design problem, in which the best set of parameters for running the computer model is determined. This is for example problems where a computer model is fitted to physical data. This design problem is related to the more general problem of predicting output from a computer model at untried inputs.

In Section 2 we consider models for computer experiments and efficient predictors, Section 3 discusses generalized least squares and implementation aspects, and in Section 4 we consider experimental design for the predictors. Section 5 is a reference manual for the toolbox, and finally examples of usage and list of notation are given in Sections 6 and 7.

2. Modelling and Prediction

Given a set of m design sites $S = [s_1 \ \cdots \ s_m]^\top$ with $s_i \in \mathbb{R}^n$ and responses $Y = [y_1 \ \cdots \ y_m]^\top$ with $y_i \in \mathbb{R}^q$. The data is assumed to satisfy the normalization conditions[1]

$$\begin{aligned} \mu[S_{:,j}] = 0, &\qquad \mathrm{V}\big[S_{:,j}, S_{:,j}\big] = 1, \quad j = 1, \ldots, n \ , \\ \mu[Y_{:,j}] = 0, &\qquad \mathrm{V}\big[Y_{:,j}, Y_{:,j}\big] = 1, \quad j = 1, \ldots, q \ , \end{aligned} \tag{2.1}$$

where $X_{:,j}$ is the vector given by the jth column in matrix X, and $\mu[\,\cdot\,]$ and $\mathrm{V}[\cdot,\cdot]$ denote respectively the mean and the covariance.

Following [9] we adopt a model $\hat{y}$ that expresses the deterministic response $y(x) \in \mathbb{R}^q$, for an n dimensional input $x \in \mathcal{D} \subseteq \mathbb{R}^n$, as a realization of a regression model $\mathcal{F}$ and a random function (stochastic process),

$$\hat{y}_\ell(x) \;=\; \mathcal{F}(\beta_{:,\ell}, x) + z_\ell(x), \quad \ell = 1, \ldots, q \ . \tag{2.2}$$

We use a regression model which is a linear combination of p chosen functions $f_j: \ \mathbb{R}^n \mapsto \mathbb{R}$,

$$\begin{aligned} \mathcal{F}(\beta_{:,\ell}, x) &= \beta_{1,\ell} f_1(x) + \cdots \beta_{p,\ell} f_p(x) \\ &= [f_1(x) \ \cdots \ f_p(x)]\, \beta_{:,\ell} \\ &\equiv f(x)^\top \beta_{:,\ell} \ . \end{aligned} \tag{2.3}$$

The coefficients $\{\beta_{k,\ell}\}$ are regression parameters.

The random process z is assumed to have mean zero and covariance

$$\mathrm{E}\big[z_\ell(w) z_\ell(x)\big] \;=\; \sigma_\ell^2\, \mathcal{R}(\theta, w, x), \quad \ell = 1, \ldots, q \tag{2.4}$$

between $z(w)$ and $z(x)$, where σ_ℓ^2 is the process variance for the ℓth component of the response and $\mathcal{R}(\theta, w, x)$ is the correlation model with parameters θ. An interpretation of the model (2.2) is that deviations from the regression model, though the response is deterministic, may resemble a sample path of a (suitably chosen) stochastic process z. In the following we will focus on the kriging predictor for y.

[1] The user does not have to think of this: The first step in the model construction is to normalize the given S, Y so that (2.1) is satisfied, see (5.1) below.

First, however, we must bear in mind that the true value can be written as

$$y_\ell(x) \;=\; \mathcal{F}(\beta_{:,\ell}, x) + \alpha(\beta_{:,\ell}, x) \ , \tag{2.5}$$

where α is the approximation error. The assumption is that by proper choice of β this error behaves like "white noise" in the region of interest, i.e., for $x \in \mathcal{D}$.

2.1. The Kriging Predictor

For the set S of design sites we have the expanded $m \times p$ design matrix F with $F_{ij} = f_j(s_i)$,

$$F \;=\; [f(s_1) \ \cdots \ f(s_m)]^\top \ , \tag{2.6}$$

with $f(x)$ defined in (2.3). Further, define R as the matrix R of stochastic-process correlations between z's at design sites,

$$R_{ij} \;=\; \mathcal{R}(\theta, s_i, s_j), \quad i, j = 1, \ldots, m \ . \tag{2.7}$$

At an untried point x let

$$r(x) \;=\; [\mathcal{R}(\theta, s_1, x) \ \cdots \ \mathcal{R}(\theta, s_m, x)]^\top \tag{2.8}$$

be the vector of correlations between z's at design sites and x.

Now, for the sake of convenience, assume that $q = 1$, implying that $\beta = \beta_{:,1}$ and $Y = Y_{:,1}$, and consider the linear predictor

$$\hat{y}(x) \;=\; c^\top Y \ , \tag{2.9}$$

with $c = c(x) \in \mathbb{R}^m$. The error is

$$\begin{aligned} \hat{y}(x) - y(x) &= c^\top Y - y(x) \\ &= c^\top (F\beta + Z) - (f(x)^\top \beta + z) \\ &= c^\top Z - z + \big(F^\top c - f(x)\big)^\top \beta \ , \end{aligned}$$

where $Z = [z_1 \ \ldots \ z_m]^\top$ are the errors at the design sites. To keep the predictor unbiased we demand that $F^\top c - f(x) = 0$, or

$$F^\top c(x) \;=\; f(x) \;. \tag{2.10}$$

Under this condition the mean squared error (MSE) of the predictor (2.9) is

$$\begin{aligned} \varphi(x) \;&=\; \mathrm{E}\big[(\hat{y}(x) - y(x))^2\big] \\ &=\; \mathrm{E}\big[(c^\top Z - z)^2\big] \\ &=\; \mathrm{E}\big[z^2 + c^\top Z Z^\top c - 2c^\top Z z\big] \\ &=\; \sigma^2\left(1 + c^\top R c - 2c^\top r\right) \;. \end{aligned} \tag{2.11}$$

The Lagrangian function for the problem of minimizing φ with respect to c and subject to the constraint (2.10) is

$$L(c,\lambda) \;=\; \sigma^2\left(1 + c^\top R c - 2c^\top r\right) - \lambda^\top(F^\top c - f) \;. \tag{2.12}$$

The gradient of (2.12) with respect to c is

$$L'_c(c,\lambda) \;=\; 2\sigma^2(Rc - r) - F\lambda \;,$$

and from the first order necessary conditions for optimality (see e.g. [7, Section 12.2]) we get the following system of equations

$$\begin{bmatrix} R & F \\ F^\top & 0 \end{bmatrix} \begin{bmatrix} c \\ \tilde{\lambda} \end{bmatrix} \;=\; \begin{bmatrix} r \\ f \end{bmatrix} \;, \tag{2.13}$$

where we have defined

$$\tilde{\lambda} \;=\; -\frac{\lambda}{2\sigma^2} \;.$$

The solution to (2.13) is

$$\begin{aligned} \tilde{\lambda} \;&=\; (F^\top R^{-1} F)^{-1}(F^\top R^{-1} r - f) \;, \\ c \;&=\; R^{-1}(r - F\tilde{\lambda}) \;. \end{aligned} \tag{2.14}$$

The matrix R and therefore R^{-1} is symmetric, and by means of (2.9) we find

$$\begin{aligned} \hat{y}(x) &= (r - F\tilde{\lambda})^\top R^{-1} Y \\ &= r^\top R^{-1} Y - (F^\top R^{-1} r - f)^\top (F^\top R^{-1} F)^{-1} F^\top R^{-1} Y \;. \end{aligned} \tag{2.15}$$

In Section 3 we show that for the regression problem

$$F\beta \;\simeq\; Y$$

the generalized least squares solution (with respect to R) is

$$\beta^* \;=\; (F^\top R^{-1} F)^{-1} F^\top R^{-1} Y \;,$$

and inserting this in (2.15) we find the predictor

$$\begin{aligned} \hat{y}(x) \;&=\; r^\top R^{-1} Y - (F^\top R^{-1} r - f)^\top \beta^* \\ &=\; f^\top \beta^* + r^\top R^{-1}(Y - F\beta^*) \\ &=\; f(x)^\top \beta^* + r(x)^\top \gamma^* \;. \end{aligned} \tag{2.16}$$

For multiple responses ($q > 1$) the relation (2.16) hold for each column in Y, so that (2.16) holds with $\beta^* \in \mathbb{R}^{p \times q}$ given by (2.15) and $\gamma^* \in \mathbb{R}^{m \times q}$ computed via the residuals, $R\gamma^* = Y - F\beta^*$.

Note that for a fixed set of design data the matrices β^* and γ^* are fixed. For every new x we just have to compute the vectors $f(x) \in \mathbb{R}^p$ and $r(x) \in \mathbb{R}^m$ and add two simple products.

Getting an estimate of the error involves a larger computational work. Again we first let $q = 1$, and from (2.11) and (2.14) we get the following expression for the MSE of the predictor,

$$\begin{aligned} \varphi(x) \;&=\; \sigma^2\left(1 + c^\top(Rc - 2r)\right) \\ &=\; \sigma^2\left(1 + (F\tilde{\lambda} - r)^\top R^{-1}(F\tilde{\lambda} + r)\right) \\ &=\; \sigma^2\left(1 + \tilde{\lambda}^\top F^\top R^{-1} F\tilde{\lambda} - r^\top R^{-1} r\right) \\ &=\; \sigma^2\left(1 + u^\top (F^\top R^{-1} F)^{-1} u - r^\top R^{-1} r\right) \;. \end{aligned} \tag{2.17}$$

where $u = F^\top R^{-1} r - f$ and σ^2 is found by means of (3.7) below. This expression generalizes immediately to the multiple response case:

for the ℓth response function we replace σ by σ_ℓ, the process variance for ℓth response function. Computational aspects are given in Section 3.1.

Remark 2.1.1. Let $x = s_i$, the ith design site. Then $r(x) = R_{:,i}$, the ith column of R, and $R^{-1}r(x) = e_i$, the ith column of the unit matrix, $e_i = I_{:,i}$. Using these relations and (2.6) in (2.16) we find

$$\begin{aligned} \hat{y}(s_i) &= f(s_i)^\top \beta^* + r(s_i)^\top R^{-1}(Y - F\beta^*) \\ &= f(s_i)^\top \beta^* + e_i^\top (Y - F\beta^*) \\ &= f(s_i)^\top \beta^* + y_i - F_{i,:}\beta^* \;=\; y_i \;. \end{aligned}$$

This shows that the Kriging predictor interpolates the design data. Further, in (2.17) we get $u = F^\top e_i - f(s_i) = 0$ and the associated MSE

$$\varphi(s_i) \;=\; \sigma^2(1 - R_{:,i}{}^\top e_i) \;=\; \sigma^2(1 - R_{ii}) \;=\; 0 \;,$$

since $R_{ii} = 1$.

Remark 2.1.2. As indicated by the name MSE (mean squared error) we expect that $\varphi(x) \geq 0$, but in (2.17) it may happen that $r^\top R^{-1} r > 1 + u^\top (F^\top R^{-1} F)^{-1} u$, in which case $\varphi(x) < 0$. This point needs further investigation, but as a first explanation we offer the following: Equation (2.11) is based on the assumption that the difference between the regression model and the true value is "white noise", and if there is significant approximation error, (2.5), then this assumption and its implications do not hold.

Remark 2.1.3. From (2.16) it follows that the gradient

$$\hat{y}' \;=\; \left[\frac{\partial \hat{y}}{\partial x_1} \;\cdots\; \frac{\partial \hat{y}}{\partial x_n}\right]^\top$$

can be expressed as

$$\hat{y}'(x) \;=\; J_f(x)^\top \beta^* + J_r(x)^\top \gamma^* \;, \tag{2.18}$$

where J_f and J_r is the Jacobian of f and r, respectively,

$$(J_f(x))_{ij} \;=\; \frac{\partial f_i}{\partial x_j}(x), \qquad (J_r(x))_{ij} \;=\; \frac{\partial \mathcal{R}}{\partial x_j}(\theta, s_i, x) \;. \tag{2.19}$$

From (2.17) it follows that the gradient of the MSE can be expressed as

$$\begin{aligned} \varphi'(x) &= 2\sigma^2\left((F^\top R^{-1}F)^{-1}u' - R^{-1}r'\right) \\ &= 2\sigma^2\left((F^\top R^{-1}F)^{-1}(F^\top R^{-1}r' - J_f\beta^*) - R^{-1}r'\right) \;. \end{aligned} \tag{2.20}$$

These expressions are implemented in the toolbox, see Section 5.2.

2.2. Regression Models

The toolbox provides regression models with polynomials of orders 0, 1 and 2. More specific, with x_j denoting the jth component of x,

Constant, $p = 1$:

$$f_1(x) \;=\; 1 \;, \tag{2.21}$$

Linear, $p = n{+}1$:

$$f_1(x) = 1, \quad f_2(x) = x_1, \quad \ldots\;, \quad f_{n+1}(x) = x_n \;, \tag{2.22}$$

Quadratic, $p = \frac{1}{2}(n{+}1)(n{+}2)$:

$$\begin{aligned} &f_1(x) = 1 \\ &f_2(x) = x_1,\; \ldots,\; f_{n+1}(x) = x_n \\ &f_{n+2}(x) = x_1^2,\; \ldots,\; f_{2n+1}(x) = x_1 x_n \\ &f_{2n+2}(x) = x_2^2,\; \ldots,\; f_{3n}(x) = x_2 x_n \\ &\cdots \qquad \cdots \qquad f_p(x) = x_n^2 \;. \end{aligned} \tag{2.23}$$

The corresponding Jacobians are (index $n{\times}q$ denotes the size of the matrix and O is the matrix of all zeros)

$$\begin{aligned} \text{constant}: &\quad J_f = [O_{n\times 1}] \;, \\ \text{linear}: &\quad J_f = [O_{n\times 1} \;\; I_{n\times n}] \;, \\ \text{quadratic}: &\quad J_f = [O_{n\times 1} \;\; I_{n\times n} \;\; H] \;, \end{aligned}$$

where we illustrate $H \in \mathbb{R}^{n\times(p-n-1)}$ by

$$n=2: \quad H = \begin{bmatrix} 2x_1 & x_2 & 0 \\ 0 & x_1 & 2x_2 \end{bmatrix} ,$$

$$n=3: \quad H = \begin{bmatrix} 2x_1 & x_2 & x_3 & 0 & 0 & 0 \\ 0 & x_1 & 0 & 2x_2 & x_3 & 0 \\ 0 & 0 & x_1 & 0 & x_2 & 2x_3 \end{bmatrix} .$$

2.3. Correlation Models

As [9] we restrict our attention to correlations of the form

$$\mathcal{R}(\theta, w, x) \;=\prod_{j=1}^{n} \mathcal{R}_j(\theta, w_j - x_j) \ ,$$

i.e., to products of stationary, one-dimensional correlations. More specific, the toolbox contains the following 7 choices

Name	$\mathcal{R}_j(\theta, d_j)$	
EXP	$\exp(-\theta_j\|d_j\|)$	
EXPG	$\exp(-\theta_j\|d_j\|^{\theta_{n+1}})$,	$0<\theta_{n+1}\le 2$
GAUSS	$\exp(-\theta_j d_j^2)$	
LIN	$\max\{0,\ 1-\theta_j\|d_j\|\}$	
SPHERICAL	$1-1.5\xi_j+0.5\xi_j^3$,	$\xi_j=\min\{1,\ \theta_j\|d_j\|\}$
CUBIC	$1-3\xi_j^2+2\xi_j^3$,	$\xi_j=\min\{1,\ \theta_j\|d_j\|\}$
SPLINE	$\varsigma(\xi_j)$, (2.24)	$\xi_j=\theta_j\|d_j\|$

Table 2.1. *Correlation functions.* $d_j = w_j - x_j$

The SPLINE correlation model is defined by

$$\varsigma(\xi_j) \;=\; \begin{cases} 1-15\xi_j^2+30\xi_j^3 & \text{for } 0\le\xi_j\le 0.2 \\ 1.25(1-\xi_j)^3 & \text{for } 0.2<\xi_j<1 \\ 0 & \text{for } \quad \xi_j\ge 1 \end{cases} \tag{2.24}$$

Some of the choices are illustrated in Figure 2.1 below. Note that in all cases the correlation decreases with $|d_j|$ and a larger value for θ_j leads to a faster decrease. The normalization (2.1) of the data implies that $|s_{ij}| \lesssim 1$ and therefore we are interested in cases where $|d_j| \lesssim 2$, as illustrated in the figure.

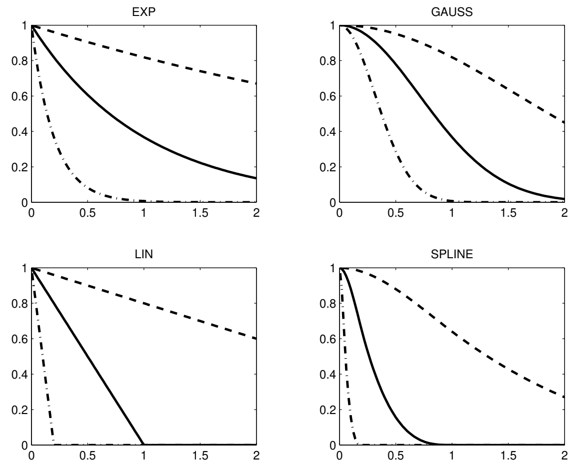

Figure 2.1. *Correlation functions for* $0 \le d_j \le 2$.
Dashed, full and dash-dotted line: $\theta_j = 0.2,\ 1,\ 5$

The correlation functions in Table 2.1 can be separated into two groups, one containing functions that have a parabolic behavior near the origin (GAUSS, CUBIC and SPLINE), and the other containing functions with a linear behaviour near the origin (EXP, LIN and SPHERICAL). The general exponential EXPG can have both shapes, depending on the last parameter: $\theta_{n+1}=2$ and $\theta_{n+1}=1$ gives the Gaussian and the exponential function, respectively.

The choice of correlation function should be motivated by the underlying phenomenon, e.g., a function we want to optimize or a physical process we want to model. If the underlying phenomenon is continuously differentiable, the correlation function will likely show a parabolic behaviour near the origin, which means that the Gaussian, the cubic or the spline function should be chosen. Conversely, physical phenomena usually show a linear behaviour near the origin, and EXP, EXPG, LIN or SPHERICAL would usually perform better, see [2]. Also note, that for large distances the correlation is 0 according to the linear, cubic, spherical and spline functions, while it is asymptotically 0 when applying the other functions.

Often the phenomenon is anisotropic. This means that different correlations are identified in different directions, i.e., the shape of the functions in Figure 2.1 differ between different directions. This is accounted for in the functions in Table 2.1, since we allow different parameters θ_j in the n dimensions.

Assuming a Gaussian process, the optimal coefficients θ^* of the correlation function solves

$$\min_{\theta}\{\ \psi(\theta)\ \equiv\ |R|^{\frac{1}{m}}\sigma^2\ \}\ , \tag{2.25}$$

where $|R|$ is the determinant of R. This definition of θ^* corresponds to maximum likelihood estimation. In Figure 2.2 we illustrate the typical behaviour of the functions involved in (2.25).

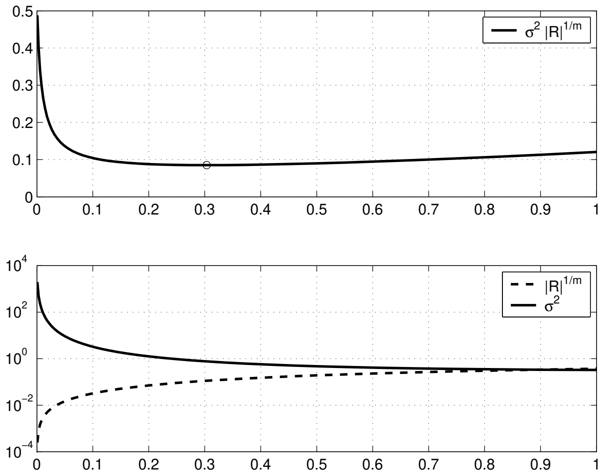

Figure 2.2. *Typical behaviour of* ψ, $|R|^{\frac{1}{m}}$ and σ^2 for $0 < \theta_j \leq 1$. *Data normalized as defined in* (5.1)

Note that $|R|^{\frac{1}{m}}$ is monotonously increasing in the interval. This is in accordance with expectation, since R_θ is close to the unit matrix for large θ, while it is indefinite for small θ. For $\theta = 0$, R is the matrix of all ones, which has rank one. In case of an indefinite R we define $\sigma^2 = \psi(\theta) = $ "∞".

See [4] for a thorough discussion of properties of different correlation models and the optimization problem (2.25).

3. Generalized Least Squares Fit

In this section we take a linear algebra view of generalized least squares estimation, and get results that are well known in statistical literature, where they are derived with probabilistic tools.

Consider an m-vector Y with outcomes of a stochastic process and let

$$y\ =\ F\beta + e\ , \tag{3.1}$$

where $F \in \mathbb{R}^{m\times p}$ (with $p < m$) is given. Assume that

$$\mathrm{E}\big[e_i\big]\ =\ 0, \qquad \mathrm{E}\big[e_i e_j\big]\ =\ \sigma^2 R_{ij}\ , \tag{3.2}$$

where R_{ij} is the $(i,j)^{\text{th}}$ element in the covariance matrix R, which is assumed to be known. Note that we can express (3.2) in matrix-vector form

$$\mathrm{E}\big[e\big]\ =\ 0, \qquad \mathrm{E}\big[ee^\top\big]\ =\ \sigma^2 R\ .$$

First, assume that the errors are uncorrelated and all have the same variance. This is equivalent with $R = I$, and the maximum likelihood estimate of the parameter vector is the least squares solution, i.e., β^* is the solution to the simple normal equations

$$(F^\top F)\beta^*\ =\ F^\top Y\ , \tag{3.3}$$

The corresponding maximum likelihood estimate of the variance is

$$\sigma^2 \;=\; \frac{1}{m}(Y - F\beta^*)^\top (Y - F\beta^*)\ . \tag{3.4}$$

To get a central estimate of the variance, the denominator m should be replaced by $m-p$, the number of degrees of freedom.

Next, assume that the errors are uncorrelated, but have different variance, i.e., $\mathrm{E}\big[e_i e_i\big] = \sigma_i^2$ and $\mathrm{E}\big[e_i e_j\big] = 0$ for $i \neq j$. Then R is the diagonal matrix

$$R \;=\; \mathrm{diag}\left(\frac{\sigma_1^2}{\sigma^2}, \ldots, \frac{\sigma_m^2}{\sigma^2}\right)\ .$$

We introduce the weight matrix W given by

$$W \;=\; \mathrm{diag}\left(\frac{\sigma}{\sigma_1}, \ldots, \frac{\sigma}{\sigma_m}\right) \quad \Leftrightarrow \quad W^2 \;=\; R^{-1}\ , \tag{3.5}$$

and the weighted observations $\tilde{Y} = WY = WF\beta + \tilde{e}$ are easily seen to satisfy

$$\mathrm{E}\big[\tilde{e}\big] \;=\; 0, \qquad \mathrm{E}\big[\tilde{e}\tilde{e}^\top\big] \;=\; W\,\mathrm{E}\big[ee^\top\big]\,W^\top \;=\; \sigma^2 I\ , \tag{3.6}$$

i.e., this transformed set of observations satisfies the assumptions for the simplest case, and it follows that β^* and σ are found by replacing F, Y in (3.3) – (3.4) by WF, WY. This results in the weighted normal equations,

$$(F^\top W^2 F)\beta^* \;=\; F^\top W^2 Y, \qquad \sigma^2 \;=\; \frac{1}{m}(Y - F\beta^*)^\top W^2 (Y - F\beta^*)\ .$$

From (3.5) we see that these relations can be expressed as

$$\begin{aligned} (F^\top R^{-1} F)\beta^* \;&=\; F^\top R^{-1} Y\ , \\ \sigma^2 \;&=\; \frac{1}{m}(Y - F\beta^*)^\top R^{-1}(Y - F\beta^*)\ . \end{aligned} \tag{3.7}$$

Finally, consider the case where the errors have nonzero correlation, i.e., R is not diagonal. For any $\alpha \in \mathbb{R}^m$ let $\eta = \alpha^\top Y$ be a linear combination of the elements in Y. Then

$$\eta \;=\; \alpha^\top F\beta + \varepsilon \quad \text{with} \quad \varepsilon = \alpha^\top e\ ,$$

and

$$\mathrm{E}\big[\varepsilon^2\big] \;=\; \mathrm{E}\big[\alpha^\top e\; e^\top \alpha\big] \;=\; \sigma^2 \alpha^\top R \alpha\ .$$

Since $\mathrm{E}\big[\varepsilon^2\big] > 0$ whenever $\alpha \neq 0$, we have shown that R is positive definite. Further, from its definition it is immediately seen that R is symmetric. These two properties imply that we can write it in factorized form,

$$R \;=\; CC^\top\ , \tag{3.8}$$

where the matrix $C^\top$ may be chosen as the Cholesky factor. As in (3.6) we see that the "decorrelation transformation"

$$\tilde{e} \;=\; C^{-1}e \;=\; C^{-1}Y - C^{-1}F\beta \;\equiv\; \tilde{Y} - \tilde{F}\beta \tag{3.9}$$

yields $\mathrm{E}\big[\tilde{e}\big] = 0$ and $\mathrm{E}\big[\tilde{e}\tilde{e}^\top\big] = \sigma^2 I$, and by similar arguments we see that (3.7) is also applicable in this general case.

3.1. Computational Aspects

The formulation (3.7) should not be used directly for practical computation if the problem is large and/or ill-conditioned. Instead β^* should be found by orthogonal transformation as the least squares solution to the overdetermined system

$$\tilde{F}\beta \;\simeq\; \tilde{Y}\ , \tag{3.10}$$

with the matrix and right-hand side obtained by solving the matrix equations

$$C\tilde{F} \;=\; F, \qquad C\tilde{Y} \;=\; Y\ .$$

The least squares solution to (3.10) can be found in the following steps,

1. Compute the "economy size" (or "thin") QR factorization of $\tilde{F}$ (see e.g. [1, Section 5.2.6]),

$$\tilde{F} \;=\; QG^\top \;, \tag{3.11}$$

where $Q \in \mathbb{R}^{m\times p}$ has orthonormal columns and $G^\top \in \mathbb{R}^{p\times p}$ is upper triangular.

2. Check that G and thereby F has full rank. If not, this is an indication that the chosen regression functions were not sufficiently linearly independent, and computation should stop. Otherwise, compute the least squares solution by back substitution in the system

$$G^\top \beta^* \;=\; Q^\top \tilde{Y} \;. \tag{3.12}$$

The auxiliary matrices can also be used to compute the process variance (3.7)

$$\sigma_\ell^2 \;=\; \frac{1}{m}\,\|\tilde{Y}_{:,\ell} - \tilde{F}\beta^*_{:,\ell}\|_2^2 \tag{3.13}$$

and the MSE (2.17),

$$\begin{aligned}\varphi_\ell(x) \;&=\; \sigma_\ell^2\left(1 + u^\top(\tilde{F}^\top\tilde{F})^{-1}u - \tilde{r}^\top\tilde{r}\right)\\ &=\; \sigma_\ell^2\left(1 + u^\top(G\,G^\top)^{-1}u - \tilde{r}^\top\tilde{r}\right)\\ &=\; \sigma_\ell^2\left(1 + \|G^{-1}u\|_2^2 - \|\tilde{r}\|_2^2\right)\end{aligned} \tag{3.14}$$

with

$$\tilde{r} \;=\; C^{-1}r, \qquad u \;=\; F^\top R^{-1} r - f \;=\; \tilde{F}^\top \tilde{r} - f \;,$$

and we have used (3.11) in the first transformation: $\tilde{F}^\top\tilde{F} = GQ^\top QG^\top = G\,G^\top$ since Q has orthonormal columns.

For large sets of design sites R will be large, and – at least with the last four choices in Table 2.1 – it can be expected to be sparse. This property is preserved in the Cholesky factor, but $R^{-1} = C^{-T}C^{-1}$ is dense.

An alternative to the Cholesky factor in (3.8) is the eigenvector-eigenvalue decomposition

$$R \;=\; V\Lambda V^\top \quad \text{with} \quad V^\top V = I, \;\; \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m) \;, \tag{3.15}$$

corresponding to $C = V\Lambda^{1/2}$, where $\Lambda^{1/2} = \mathrm{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m})$; all the λ_j are real and positive. However, this factorization is more costly to compute, and the eigenvector matrix V is often a dense matrix.

Depending on the choice of $\mathcal{R}$ and the parameters θ the matrix R may be vey ill-conditioned. This is investigated in [4, sections 4-5], and in order to reduce effects of rounding errors the implementation uses a modified Cholesky factorization, where (3.8) is replaced by

$$CC^\top \;=\; R + \mu I \quad \text{with} \quad \mu = (10{+}m)\varepsilon_{\mathsf{M}} \;, \tag{3.16}$$

where ε_{M} is the socalled machine accuracy (or unit round-off), $\varepsilon_{\mathsf{M}} = 2^{-52} \simeq 2.22 \cdot 10^{-16}$ in MATLAB.

4. Experimental Design

Experimental design arises in this context in deciding how to select the inputs at which to run the deterministic computer code in order to most efficiently control or reduce the statistical uncertainty of the computed prediction. This section introduces two algorithms with "space filling" properties. Note that Latin Hypercube designs are based on random numbers, and the other algorithm produces deterministic designs. See [6], [8], [9] or [10] for further discussion and more advanced designs.

4.1. Rectangular Grid

Assume that the region $\mathcal{D} \in \mathbb{R}^n$ under interest is a box, defined by $\ell_j \le x_j \le u_j$, $j = 1, \ldots, n$. The simplest distribution of design sites is defined by all different combinations of

$$s_j^{(i)} = \ell_j + k_j^{(i)} \frac{u_j - \ell_j}{\nu_j}, \quad k_j^{(i)} = 0, 1, \ldots, \nu_j \ ,$$

where the $\{\nu_j\}$ are integers. If all $\nu_j = \nu$, then the number of these design points is $(\nu+1)^n$.

4.2. Latin Hypercube Sampling

Latin hypercube sampling, due to McKay et al. [5], is a strategy for generating random sample points ensuring that all portions of the vector space is represented. Consider the case where we wish to sample m points in the n dimensional vector space $\mathcal{D} \in \mathbb{R}^n$. The Latin hypercube sampling strategy is as follows:

1. Divide the interval of each dimension into m non-overlapping intervals having equal probability (here we consider a uniform distribution, so the intervals should have equal size).
2. Sample randomly from a uniform distribution a point in each interval in each dimension.
3. Pair randomly (equal likely combinations) the point from each dimension.

5. Reference Manual

This section is a presentation of the functions in the DACE toolbox. The contents are

5.1. Model Construction	
`dacefit`	Find the DACE model to a given set of design data and given regression and correlation models

5.2. Evaluate the Model	
`predictor`	Use the DACE model to predict the function at one or more untried sites

5.3. Regression Models	
`regpoly0`	Zero order ploynomial
`regpoly1`	First order ploynomial
`regpoly2`	Second order ploynomial

5.4. Correlation Models	
`correxp`	Exponential
`correxpg`	Generalized exponential
`corrgauss`	Gaussian
`corrlin`	Linear
`corrspherical`	Spherical
`corrspline`	Cubic spline

5.5. Experimental Design	
`gridsamp`	Design sites in a rectangular grid
`lhsamp`	Latin hypercube distributed random points

5.6. Auxiliary Functions	
`dsmerge`	Merge data for multiple design sites

5.7. Data Files	
`data1.mat`	Example data S and Y

In the following we give details of the MATLAB functions. The file `data1.mat` is used in the first example of Section 6.

Installation: Obtain the archive dace.zip containing the software at the web-site http://www.imm.dtu.dk/∼hbn/dace
Unzip the archive at a convenient place. On UNIX systems use the command `unzip dace.zip`, on Windows systems use WinZip or a similar program.

The archive contains a folder named dace containing the software and this documentation. The path to the software (i.e., to the dace folder) should be included in the MATLAB search path. Use either the `pathtool` or the `addpath` command. See the MATLAB manual for further directions.

5.1. Model Construction

Purpose: Find the DACE model to a given set of design data and given regression and correlation models.

Call:

```
[dmodel, perf] = dacefit(S, Y, regr, corr, theta0)
[dmodel, perf] = ...
    dacefit(S, Y, regr, corr, theta0, lob, upb)
```

Input parameters:

`S`	Design sites: an $m \times n$ array with $\mathtt{S(i,:)} = s_i^\top$.
`Y`	$m \times q$ array with responses at `S`.
`regr`	Handle to a function like (5.2) below.
`corr`	Handle to a function like (5.3) below.
`theta0`	If `lob` and `upb` are not present, then `theta0` should hold the correlation function parameters, θ. Otherwise `theta0` should hold initial guess on θ. See Section 5.4 about permissible lengths of `theta0`.
`lob,upb`	Optional. If present, then they should be vectors of the same length as `theta0` and should hold respectively lower and upper bounds on θ. If they are not present, then θ is given by the values in `theta0`.

Output:

`dmodel` DACE model. Struct with the elements

`regr`	handle to the regression function,
`corr`	handle to the correlation function,
`theta`	correlation function parameters,
`beta`	generalized least squares estimate, β^* in (2.16)
`gamma`	correlation factors, γ^* in (2.16),
`sigma2`	estimate of the process variance σ^2,
`S`	scaled design sites, see (5.1) below,
`Ssc`	$2 \times n$ array with scaling factors for design sites,
`Ysc`	$2 \times q$ array with scaling factors for design responses,
`C`	Cholesky factor of correlation matrix, (3.16),
`Ft`	decorrelated regression matrix, $\tilde{F}$ in (3.10),
`G`	matrix G from (3.11).

`perf` Information about the optimization.
Struct with the elements

`nv`	Number of evaluations of objective function (2.25) used to find θ^*,
`perf`	Array with `nv` columns with current values of $[\theta;\ \psi(\theta);\ \mathit{type}]$. $\lvert\mathit{type}\rvert = 1$, 2 or 3, indicate "start", "explore" or "move". A negative value for *type* indicates an uphill trial step.

Remarks. The first step in `dacefit` is to normalize the input so that (2.1) is satisfied,

```
mS = mean(S);  sS = std(S);
for j=1:n, S(:,j) = (S(:,j) - mS(j))/sS(j); end
mY = mean(Y);  sY = std(Y);
for j=1:q, Y(:,j) = (Y(:,j) - mY(j))/sY(j); end
```
(5.1)

The values in `mS` and `sS` are returned in `dmodel.Ssc`, and `dmodel.Ysc` = `[mY; sY]`.

All computation is performed on the normalized data, but the process variance is for the original data, $(\texttt{dmodel.sigma2})_j = \texttt{sY}_j^2 \cdot \overline{\sigma}_j^2$, where $\overline{\sigma}_j^2$ is the estimator for the jth column of the normalized responses.

The matrices R and C are stored in sparse format, and it is exploited that we only need to store the upper triangle of the symmetric matrix R.

As indicated in Section 2.3, the determination of the optimal correlation parameters θ^* is an optimization problem with box constraints, $\ell_j \le \theta_j \le u_j$. We have developed a simple but efficient algorithm with successive coordinate search and pattern moves, as in the Hooke & Jeeves method, see e.g. [3, Section 2.4]. Details are given in [4, Section 6]. The objective function $\psi(\theta) \equiv |R|^{\frac{1}{m}} \sigma^2$ was presented in [9] for the case $q=1$. In the multiple response case we let $\sigma^2 := \sigma_1^2 + \cdots + \sigma_q^2$, with each component of σ^2 computed by (3.13).

5.2. Evaluate the Model

Purpose: Use the DACE model to predict the function at one or more untried sites.

Call:

```
y = predictor(x, dmodel)
[y, or] = predictor(x, dmodel)
[y, dy, mse] = predictor(x, dmodel)
[y, dy, mse, dmse] = predictor(x, dmodel)
```

Input parameters:

`x`	m trial site(s) with n dimensions. If $m=1$ and $n>1$, then both a row and a column vector is accepted. Otherwise `x` must be an $m \times n$ array with the sites stored rowwise.
`dmodel`	Struct with the DACE model, see Section 5.1.

Output:

`y`	Predicted response, `y(i)` $= \hat{y}($`x(i,:)`$)$, see (2.16).

`or`	Optional result. Depends on the number of sites, $m = 1$: gradient $\hat{y}'$, (2.18). Column vector with n elements, $m > 1$: m-vector with estimated MSE, (3.14).
`dy`	Optional result, allowed only if $m=1$: $n \times q$ array with Jacobian of $\hat{y}$ (gradient $\hat{y}'(\mathtt{x})$ (2.18) if $q=1$).
`mse`	Optional result, allowed only if $m=1$: estimated MSE, (3.14).
`dmse`	Optional result, allowed only if $m=1$: $n \times q$ array with Jacobian of φ (gradient $\varphi'(\mathtt{x})$ (2.20) if $q=1$).

Remarks. The computation is performed on normalized trial sites, cf. (5.1), but the returned results are in the original "units".

The special treatment of optional results when there is only one trial site was made so that `predictor` can be used as the objective function in a Matlab optimization function demanding the gradient.

5.3. Regression Models

The toolbox provides functions that implement the polynomials discussed in Section 2.2. All of these conform with the specifications given in (5.2) at the end of this section.

Purpose: Get values of zero order polynomial, (2.21).

Call:

```
f = regpoly0(S)
[f, df] = regpoly0(S)
```

Input parameter:

`S`	$m \times n$ matrix with design sites stored rowwise.

Output:

`f`	$m \times 1$ vector with all ones.
`df`	Optional result, Jacobian for the first site: $n \times 1$ vector with all zeros.

Purpose: Get values of first order polynomials, (2.22).

Call:
```
f = regpoly1(S)
[f, df] = regpoly1(S)
```

Input parameter:

`S`	$m \times n$ matrix with design sites stored rowwise.

Output:

`f`	$m \times (n+1)$ matrix with `f(i,j)` $= f_j(x_i)$.
`df`	Optional result, Jacobian for the first site, Section 2.2.

Purpose: Get values of second order polynomials, (2.23).

Call:
```
f = regpoly2(S)
[f, df] = regpoly2(S)
```

Input parameter:

`S`	$m \times n$ matrix with design sites stored rowwise.

Output:

`f`	$m \times p$ matrix with `f(i,j)` $= f_j(x_i)$; $p = \frac{1}{2}(n+1)(n+1)$.
`df`	Optional result, Jacobian for the first site, Section 2.2.

Remark. The user may supply a new regression model in the form of a function that must have a declaration of the form

```
function [f, df] = regress(S)
```
(5.2)

For a given set of m sites `S` with the ith site stored in `S(i,:)` it should return the $m \times p$ matrix `f` with elements `f(i,j)` $= f_j($`S(i,:)`$)$, cf. (2.3). If it is called with two output arguments, then the Jacobian J_f, (2.19), should be returned in `df`. This option is needed only if the predictor is called with the option of returning also the gradient.

5.4. Correlation Models

The toolbox provides seven functions that implement the models presented in Table 2.1, see the list on page 18. All of these conform with the specifications given in (5.3) at the end of this section. We only present one of them in detail,

Purpose: Get values of the function denoted EXP in Table 2.1.

Call:
```
r = correxp(theta, d)
[r, dr] = correxp(theta, d)
```

Input parameters:

`theta`	Parameters in the correlation function. A scalar value is allowed. This corresponds to an isotropic model: all θ_j equal to `theta`. Otherwise, the number of elements in `theta` must equal the dimension n given in `d`.
`d`	$m \times n$ array with differences between sites.

Output:

`r`	Correlations, `r(i)` $= \mathcal{R}(\theta,$ `d(i,:)`$)$.
`dr`	Optional result. $m \times n$ array with the Jacobian J_r, (2.19).

Remarks. The Jacobian is meaningful only when `d` holds the differences between a point x and the design sites, as given in S, $d_{i,:} = x^\top - S_{i,:}$. The expression given in Table 2.1 can be written as

$$r_i = \prod_{j=1}^{n} \exp(-\theta_j |d_{ij}|) = \exp\left(\sum_{j=1}^{n} -\theta_j \tau_{ij} (x_j - S_{ij})\right) ,$$

where τ_{ij} is the *sign* of $d_j^{(i)}$. The corresponding Jacobian is given by

$$(J_r)_{ij} = -\theta_j \cdot \tau_{ij} \cdot r_i \ .$$

Remarks. The handling of isotropic models is similar in `corrgauss`, `corrgauss`, `corrlin`, `corrspherical` and `corrspline`, while `correxpg` needs special treatment of the exponent θ_{n+1}. Here `theta` must be a vector with either $n+1$ or 2 elements. In the latter case $\theta_j =$ `theta(1)`, $j = 1, \ldots, n$ and $\theta_{n+1} =$ `theta(2)`.

The user may supply a new correlation model in the form of a function that must have a declaration of the form

```
function [r, dr] = corr(theta, d)
```
(5.3)

For a given set of parameters `theta` $= \theta$ and differences `d(i,:)` $= x^\top - S_{i,:}$ it should return the column vector `r` with elements `r(i)` $= \mathcal{R}(\theta, x, S_{i,:})$. If it is called with two output arguments, then the Jacobian J_r, (2.19), should be returned in `dr`. This option is needed only if the predictor is called with the option of returning also the gradient.

5.5. Experimental Design

Currently the toolbox provides implementations of the two designs of Section 4:

Purpose: Get design sites in a rectangular grid, Section 4.1.

Call: `X = gridsamp(range, q)`

Input parameters:

`range`	$2{\times}n$ with lower and upper limits, `range(:,j)` $= [\ell_j \; u_j]^\top$.
`q`	Vector with `q(j)` holding the number of intervals in the jth direction. If `q` is a scalar, then it uses this number in all directions.

Output:

`X`	$m{\times}n$ array with grid points.

Purpose: Compute latin hypercube distributed random points, Section 4.2.

Call:
```
S = lhsamp
S = lhsamp(m)
S = lhsamp(m, n)
```

Input parameters:

`m`	Number of sample points to generate. If not present, then `m`=1.
`n`	Number of dimensions. If not present, then `n`=`m`.

Output:

`S`	$m{\times}n$ matrix with the generated sample points chosen from uniform distributions on `m` subdivisions of the interval $]0.0, 1.0[$.

5.6. Auxiliary Functions

The Kriging model presumes distinct design sites, and the toolbox provides a function that can compress the data if this condition is not satisfied.

Purpose: Merge data for multiple design sites.

Call:
```
[mS, mY] = dsmerge(S, Y)
[mS, mY] = dsmerge(S, Y, ds)
[mS, mY] = dsmerge(S, Y, ds, nms)
[mS, mY] = dsmerge(S, Y, ds, nms, wtds)
[mS, mY] = dsmerge(S, Y, ds, nms, wtds, wtdy)
```

Input parameters:

`S`	$m{\times}n$ array with design sites stored rowwise.
`Y`	$m{\times}q$ array with responses at `S`.
`ds`	Threshold for equal, normalized sites. Default is `1e-14`.
`nms`	Norm, in which the distance is measured.

nms = 1 1-norm (sum of absolute coordinate differences)
2 2-norm (Euclidean distance) (default)

wtds What to do with the S-values in case of multiple points.
wtds = 1 return the mean value (default)
2 return the median value
3 return the 'cluster center'

wtdy What to do with the Y-values in case of multiple points.
wtdy = 1 return the mean value (default)
2 return the median value
3 return the 'cluster center' value
4 return the minimum value
5 return the maximum value

Output:

mS Compressed design sites, with multiple points merged according to wtds.

mY Responses, compressed according to wtdy.

5.7. Data Files

Currently the toolbox contains one test data set, illustrated in Section 6. The command

```
load data1
```

makes the arrays $\mathtt{S} \in \mathbb{R}^{75\times 2}$ and $\mathtt{Y} \in \mathbb{R}^{75\times 1}$ available in the workspace; i.e., $m=75$, $n=2$, $q=1$. The design sites stored in S are sampled in the two-dimensional area $[0,100]^2$.

6. Examples of Usage

6.1. Work-through Example

This example demonstrates simple usage of the two most important functions in the toolbox, namely `dacefit` and `predictor`. The example shows how you can obtain a surface approximation and corresponding error estimates for the approximation to a given data set. The example also shows how gradient approximations at given points can be obtained for both predictor and error estimate.

We start by loading the data set `data1.mat` provided with the toolbox, cf. Section 5.7,

```
load data1
```

Now the 75×2 array S and 75×1 array Y are present in the workspace.

We choose the POLY0 regression function and the GAUSS correlation function. Assuming anisotropy we choose the following starting point and bounds for θ

```
theta = [10 10];  lob = [1e-1 1e-1];   upb = [20 20];
```

We are now ready to make the model by calling `dacefit`,

```
[dmodel, perf] = ...
     dacefit(S, Y, @regpoly0, @corrgauss, theta, lob, upb)
```

From the returned results we can extract information about the generated model. The number of evaluations of the objective function (2.25) to find θ^* and the values of θ^* are

```
perf.nv = 15
dmodel.theta = [3.5355   2.1022]
```

The generalized least squares estimate β^* and the estimated process variance σ^2 are

```
dmodel.beta = 0.0918
dmodel.sigma2 = 3.3995
```

Having the model stored in the structure array `dmodel` we may use it for prediction at new (untried) sites. We generate a grid of points on which to evaluate the predictor (2.16). We choose a $40{\times}40$ mesh of points distributed equidistantly in the area $[0, 100]^2$ covered by the design sites, cf. Section 5.7, and call the kriging predictor with the mesh points and the `dmodel`,

```
X = gridsamp([0 0;100 100], 40);
[YX MSE] = predictor(X, dmodel);
```

The returned vector `YX` of predicted values at the mesh and `MSE` the mean squared error for each predicted point. To plot the results we reshape the coordinate matrix and `YX` to match the grid

```
X1 = reshape(X(:,1),40,40);   X2 = reshape(X(:,2),40,40);
YX = reshape(YX, size(X1));
```

Mesh plot of the predicted values at the grid points, and add the design sites

```
figure(1),  mesh(X1, X2, YX)
hold on,
plot3(S(:,1),S(:,2),Y,'.k', 'MarkerSize',10)
hold off
```

The resulting plot is shown in Figure 6.1.

Next, to get a mesh plot of the mean squared error in a new figure window we issue the commands

```
figure(2),  mesh(X1, X2, reshape(MSE, size(X1)) )
```

The resulting plot is shown in Figure 6.2. From the Figure we note how areas with few design sites (e.g. the center area) have high MSE values.

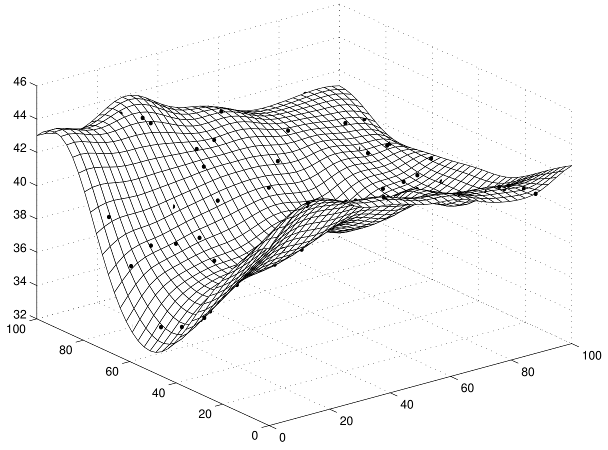

Figure 6.1. *Predicted values*

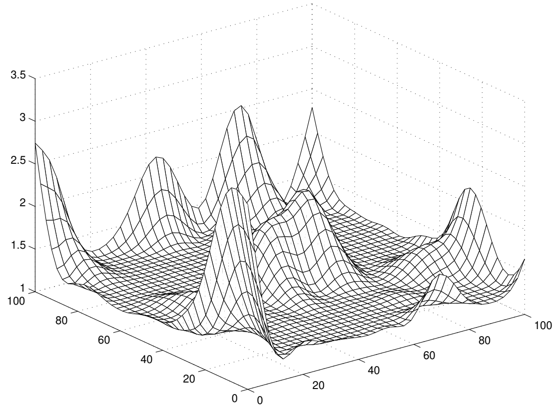

Figure 6.2. *Mean squared error*

The `predictor` function also allows for prediction of gradients provided a single point, e.g. here is how to predict the gradient at the first design site,

```
[y, dy] = predictor(S(1,:), dmodel)
y =  34.1000
dy = 0.2526
     0.1774
```

The gradient of MSE is also available, e.g. the for the point $(50, 50)$,

```
[y, dy, mse, dmse] = predictor([50 50], dmodel)
y = 38.0610
dy = -0.0141
     -0.0431
mse = 0.7526
dmse = 0.0004
       0.0187
```

6.2. Adding a Regression Function

This example shows how a user provided regression function can be added to the toolbox. Adding a correlation function is done in virtually the same way, for which reason a specific example of such is not given here.

As noted in the reference manual in Section 5, the regression functions and the correlation functions must be implemented with a specific interface. Below is an example on how to implement a reduced (without the cross-dimensional products) second order polynomial regression function $f(x) = [1 \; x_1 \; \ldots \; x_n \; x_1^2 \; \ldots \; x_n^2]^\top$.

```
function  [f, df] = regpoly2r(S)
%REGPOLY2R  Reduced 2nd order polynomial regr. function
% Call:    [f, df] = regpoly2(S)
% S : m*n matrix with design sites
% f =  [1  S  S^2]
% df is the Jacobian at the first point (first row in S)

[m n] = size(S);
f = [ones(m,1)  S  S.^2];

if  nargout > 1
  df = [zeros(n,1)  eye(n)  2*diag(S(1,:))];
end
```


Note that the array `f` is $m \times p$ with $p = 1+2n$. The last part with the gradient calculation is needed only if the gradient feature of the `predictor` function is used.

7. Notation

$m, \; n$	number of design sites and their dimensionality
p	number of basis functions in regression model
q	dimensionality of responses
$\mathcal{F}(\beta, x)$	regression model, $\mathcal{F}(\beta, x) = f(x)^\top \beta$, see Section 2
$\mathcal{R}(\theta, w, x)$	correlation function, see Section 2.3
C	factorization (e.g. Cholesky) of R, $\; R = C^T C$
$\mathrm{E}[\cdot]$	expectation operator
f_j	basis function for regression model
f	p-vector, $\; f(x) = [f_1(x) \; \cdots \; f_p(x)]^\top$
F	expanded design $m \times p$-matrix, see (2.6)
$\tilde{F}, \; \tilde{Y}$	transformed data, see (3.9)
R	$m \times m$-matrix of stochastic-process correlations, see (2.7)

r	m-vector of correlations, see (2.8)
S	$m \times n$ matrix of design sites, see Section 2
$S_{i,:}$, $S_{:,j}$	ith row and jth column in S, respectively
s_i	ith design site, vector of length n. $s_i^\top = S_{i,:}$
$\mathrm{V}[w,x]$	covariance between w and x (2.4)
w, x	n-dimensional trial points
x_j	jth component in x
Y	$m \times q$-matrix of responses, see Section 2
y_i	response at ith design site, q-vector
$\hat{y}$	predicted response, see (2.2)
z	q-dimensional stochastic process, see (2.2)
β	$p \times q$-matrix of regression parameters, see (2.2), (2.16)
γ	$m \times q$-matrix of correlation constants, see (2.16)
θ	parameters of correlation model, q-vector
σ^2	process variance (of z), see (2.4)
$\varphi(x)$	mean squared error of $\hat{y}$, see (2.11), (3.14)

References

[1] G. Golub, C. Van Loan, *Matrix Computations.* Johns Hopkins University Press, Baltimore, USA, 3rd edition, 1996.

[2] E.H. Isaaks, R.M. Srivastava, *An Introduction to Applied Geostatistics.* Oxford University Press, New York, USA, 1989.

[3] J. Kowalik, M.R. Osborne, *Methods for Unconstrained Optimization Problems.* Elsevier, New York, USA, 1968.

[4] S.N. Lophaven, H.B. Nielsen, J. Søndergaard, *Aspects of the Matlab Toolbox DACE.* Report IMM-REP-2002-13, Informatics and Mathematical Modelling, DTU. (2002), 44 pages. Available as http://www.imm.dtu.dk/∼hbn/publ/TR0213.ps

[5] M.D. McKay, W.J. Conover, R.J. Beckman, *A comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code*, Technometrics, vol. 21, no. 2, 1979.

[6] W. G. Müller, *Collecting Spatial Data. Optimum Design of Experiments for Random Fields.* Physica-Verlag, Heidelberg, Germany, 2001.

[7] J. Nocedal, S.J. Wright, *Numerical Optimization*, Springer, New York, USA, 1999.

[8] J.A. Royle, D. Nychka, *An Algorithm for the Construction of Spatial Coverage Designs with Implementation in Splus*, Computers and Geosciences, vol. 24, no. 5, pp. 479-488, 1998.

[9] J. Sacks, W.J. Welch, T.J. Mitchell, H.P. Wynn, *Design and Analysis of Computer Experiments*, Statistical Science, vol. 4, no. 4, pp. 409-435, 1989.

[10] T.W. Simpson, J.D. Peplinski, P.N. Koch, J.K. Allen, *Metamodels for Computer-Based Engineering Design: Survey and Recommendations* Engineering with Computers, vol. 17, pp. 129-150, 2001.