Pivoting Cholesky Decomposition applied to Emulation and Validation of computer models

Leonardo S. Bastos* Anthony O'Hagan

University of Sheffield, UK

June, 2007

Abstract

Estimation and prediction of computer models using Gaussian processes emulators require evaluation of the determinant and inverse of large matrices. The Cholesky decomposition provides the most numerically stable method of calculating these quantities. In Gaussian processes, highly correlated points can lead to instability in the Cholesky decomposition, because the associated variance matrix is numerically ill-conditioned. This work describes a variation of the Pivoting Cholesky Decomposition for ill-conditioned matrices. This method is also used to build validation diagnostics of a Gaussian process emulator.

1 Introduction

Computer models, also called simulators, have been used to represent real systems in almost all fields of science and technology. Some real systems can be theoretically represented by abstract complex mathematical models, and those models are implemented in a computer in

*Corresponding author. Department of Probability and Statistics, University of Sheffield, Sheffield, S3 7RH, UK *Email address*: l.bastos@shef.ac.uk.

order to simulate real-world systems. Computer models have been used to investigate real-world systems where physical observations are either highly expensive or too time-consuming to obtain. (Sacks et al. (1989), Welch et al. (1992), Santner et al. (2003)).

Simulators are usually deterministic models, deterministic in its sense that their outputs are always the same for the same inputs. However, the values of the outputs of the model are unknown, and from a Bayesian perspective the uncertainty of the simulator can be expressed by a stochastic process. Using a Gaussian process to represent our judgements about the simulator yields a Gaussian Process Emulator. Welch et al. (1992) present an application of Gaussian Process to screening (input selection) and prediction in computer models. Haylock and O'Hagan (1994) use Gaussian process emulator to make inference for the output when there is uncertainty on the inputs. O'Hagan, Kennedy, and Oakley (1998) and Oakley and O'Hagan (2002) present an improved uncertainty analysis for computer model outputs. Oakley and O'Hagan (2004) present a sensitivity analysis for the Gaussian process emulator where they studied how an output of a model responds to changes in the model inputs. O'Hagan (2006) presents a tutorial of statistical analysis in computer models.

Inference and prediction using Gaussian process emulators require a considerable number of evaluations of determinant and inverses of matrices, Harville (1997) shows that the Cholesky method is the most numerically stable method of calculating these quantities. The Cholesky method can also be applied in design, sensitivity analysis, calibration and validation of computer models. In this work, the Cholesky method is used to build validation diagnostics for Gaussian Process Emulators, where predictive variance matrices are decomposed. However, highly correlated data can lead to ill-conditioned matrices, invalidating not only the Cholesky method, but also other methods for positive definite matrices, e.g. LDL factorization, (Golub and Loan, 1996).

The purpose of this work is to present validation diagnostics based on the Pivoting Cholesky decomposition of the predictive variance matrix. The Cholesky decomposition is reviewed in Section 2.1, the Pivoting Cholesky decomposition and a modified version of it for positive definite ill-conditioned matrices are presented in Section 2.2. The theory of Gaussian Process Emulators is given in Section 3.1, where the use of the Cholesky decomposition is indicated. Validation of computer models, particularly for Gaussian Process Emulators, is described on Section 3.2, where also some possible diagnostics are presented. A two dimentional toy example and a real data example of the validation diagnostics using the Pivoting Cholesky decomposition are illustrated in Section 4, and Section 5 presents some conclusions.

2 The Cholesky Method

The Cholesky decomposition was developed by the French mathematician André-Louis Cholesky, and published after his death by Benoît (1924). The method received little attention after its publication. However, the Cholesky decomposition was analysed by Fox, Huskey, and Wilkinson (1948) and, in the same year, Turing (1948) presented a result on stability of the method. After that, the Cholesky decomposition became very popular. See more details about historical context in Taussky and Todd (2006) and Brezinski (2006).

2.1 Simple Cholesky Decomposition

Matrices are denoted in this work by bold capital letters, $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$, and submatrices are specified with the colon notation, as used in Golub and Loan (1996). $\mathbf{A}(p{:}q, r{:}s)$ denotes the submatrix of $\mathbf{A}$ formed by the intersection of rows p to q and columns r to s. Particular cases: $\mathbf{A}(i, j)$ denotes the element in row i and column j, $\mathbf{A}(i, :)$ denotes the row i, and $\mathbf{A}(:, j)$ the column j.

Formally, the Cholesky method decomposes a symmetric positive definite matrix $\mathbf{A}$ uniquely into a product of a lower triangular matrix $\mathbf{L}$ and its transpose, i.e. $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, or equivalently $\mathbf{A} = \mathbf{R}^T\mathbf{R}$ where $\mathbf{R}$ is a upper triangular matrix. Without loss of generality the presented Cholesky decomposition in this work finds an upper triangular matrix, $\mathbf{R}$. This method is the recursive process described in Algorithm 1, where basically at each step k the line k of matrix $\mathbf{R}$ is calculated and the matrix $\mathbf{A}(k : n, k : n)$ is updated and used in a recursive process.

Algorithm 1 This algorithm computes the Cholesky decomposition $\mathbf{A} = \mathbf{R}^T\mathbf{R}$ of a symmetric positive definite matrix $\mathbf{A} \in \Re^{n \times n}$

$\mathbf{R} = \mathbf{0}$ {define a $n \times n$ zero matrix}
for $k = 1$ to n **do**
  $\mathbf{R}(k, k) = \sqrt{\mathbf{A}(k, k)}$
  $\mathbf{R}(k, k+1 : n) = \mathbf{R}(k, k)^{-1}\mathbf{A}(k, k+1 : n)$
  $\mathbf{A}(k+1 : n, k+1 : n) = \mathbf{A}(k+1 : n, k+1 : n) - \mathbf{R}(k, k+1 : n)^T\mathbf{R}(k, k+1 : n)$
end for

Using the Cholesky decomposition the determinant of $\mathbf{A}$ is given by the square of the product of the diagonal values of matrix $\mathbf{R}$, $det(\mathbf{A}) = \prod_{i=1}^{n} \mathbf{R}(i, i)^2$, and the inverse is

obtained by $(\mathbf{A})^{-1} = (\mathbf{R}^T\mathbf{R})^{-1} = \mathbf{R}^{-1}(\mathbf{R}^{-1})^T$, where $\mathbf{R}^{-1}$ is calculated by the backward substitution method. The Cholesky decomposition is often used to help solving the linear system $\mathbf{A}x = b$, when $\mathbf{A}$ is symmetric positive definite. For example, the equations of the linear least square problem are of this form, Gentle (1998).

An arbitrary permutation of rows and columns of matrix $\mathbf{A}$ can be decomposed by the Cholesky algorithm, $\mathbf{P}^T\mathbf{A}\mathbf{P} = \mathbf{R}^T\mathbf{R}$, where $\mathbf{P}$ is a permutation matrix and $\mathbf{R}$ is an upper triangular matrix. The permutation matrix is an orthogonal matrix, so the matrix $\mathbf{A}$ can be rewritten as $\mathbf{A} = (\mathbf{R}\mathbf{P}^T)^T\mathbf{R}\mathbf{P}^T$, so that $\mathbf{R}\mathbf{P}^T$ is a decomposition of $\mathbf{A}$. Although it is not the Cholesky decomposition of $\mathbf{A}$, because it is not triangular, it can be computed quickly using a simple modification of Algorithm 1.

Notice that, the determinant of $\mathbf{A}$ is the same as the determinant of its symmetric permutation $\mathbf{P^T}\mathbf{A}\mathbf{P}$, i.e. $det(\mathbf{A}) = det(\mathbf{P^T}\mathbf{A}\mathbf{P})$, but in order to find the inverse of $\mathbf{A}$, the following relationship using the inverse of its symmetric permutation can be used $(\mathbf{A})^{-1} = \mathbf{P^T}(\mathbf{P^T}\mathbf{A}\mathbf{P})^{-\mathbf{1}}\mathbf{P}$, where $(\mathbf{P^T}\mathbf{A}\mathbf{P})^{-\mathbf{1}} = \mathbf{R}^{-1}(\mathbf{R}^{-1})^T$.

2.2 The Pivoting Cholesky Decomposition

If $\mathbf{A} \in \Re^{n\times n}$ is symmetric positive semidefinite, of rank $m < n$, Higham (2002, Thm. 10.9) shows that there exists a permutation matrix $\mathbf{P} \in \Re^{n\times n}$ such that

$$\mathbf{P^T}\mathbf{A}\mathbf{P} = \mathbf{R}^T\mathbf{R}, \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{1}$$

where $\mathbf{R}_{11}$ is an $m \times m$ upper triangular matrix with positive diagonal elements, and $\mathbf{R}_{12}$ is $m \times (n-m)$ matrix.

The process of finding $\mathbf{R}$ and $\mathbf{P}$ for symmetric positive semidefinite matrices is called Pivoting Cholesky decomposition (PCD). If the rank of $m = n$, the PCD reduces to a simple Cholesky decomposition of a symmetric permutation of $\mathbf{A}$. The PCD is an extension of Cholesky decomposition where a pivoting step and a stopping criterion are included. The pivoting step consists in finding, at each iteration k, the largest possible element of the diagonal of the updated matrix $\mathbf{A}(k{:}n, k{:}n)$, the index of the largest element is saved and called the pivot. And the stopping criterion consists in stopping the algorithm at iteration k when the the estimated rank of $\mathbf{A}$ is k. The PCD algorithm, which is an extension of the Cholesky decomposition Algorithm 1, is described on Algorithm 2.

Algorithm 2 This algorithm computes the Pivoting Cholesky decomposition $\mathbf{P^T A P} = \mathbf{R}^T\mathbf{R}$ of a symmetric positive semidefinite matrix $\mathbf{A} \in \Re^{n\times n}$. The nonzero elements of the permutation matrix $\mathbf{P}$ are given by $\mathbf{P}(piv(k), k) = 1$, $k = 1, \ldots, n$

$\mathbf{R} = \mathbf{0}$ {define a $n \times n$ zero matrix}
$piv = 1 : n$
for $k = 1$ to n **do**
  $q = \{i : \mathbf{A}(i,i) = \max(diag(\mathbf{A}(k : n, k : n)))\} + k - 1$ {Finding the pivot}
  if Stopping criterion **then**
    stop {rank of $\mathbf{A}$ is $(k-1)$}
  end if
  $\mathbf{A}(:,k) \leftrightarrows \mathbf{A}(:,q)$ {Swap columns}
  $\mathbf{R}(:,k) \leftrightarrows \mathbf{R}(:,q)$ {Swap columns}
  $\mathbf{A}(k,:) \leftrightarrows \mathbf{A}(q,:)$ {Swap rows}
  $piv(k) \leftrightarrows piv(q)$ {Swap pivoting position}
  $\mathbf{R}(k,k) = \sqrt{\mathbf{A}(k,k)}$
  $\mathbf{R}(k, k+1 : n) = \mathbf{R}(k,k)^{-1}\mathbf{A}(k, k+1 : n)$
  $\mathbf{A}(k+1 : n, k+1 : n) = \mathbf{A}(k+1 : n, k+1 : n) - \mathbf{R}(k, k+1 : n)^T\mathbf{R}(k, k+1 : n)$
end for

Theoretically the algorithm could be stopped when the current pivot is equal to zero, but in practice the pivots may be wrongly derived due to rounding errors. For the PCD, Hammarling et al. (2007) present an analysis of the efficiency of the PCD for different stopping criteria, and they propose a stopping criterion which is more reliable to detect the rank. The proposed stopping criterion is to set the rank of $\mathbf{A}$ as k if the following inequality is true

$$\max_{k\le i\le n} \mathbf{A}^{(k)}(i,i) \le \epsilon \max_{k\le i\le n} \mathbf{A}^{(0)}(i,i), \tag{2}$$

where $\max_{k\le i\le n} \mathbf{A}^{(k)}(i,i)$ is the current pivot at iteration k, $\mathbf{A}^{(k)}$ is the updated matrix after $(k-1)$ iterations, $\mathbf{A}^{(0)} = \mathbf{A}$, $\epsilon = nu$, and u is the *unit roundoff* (or machine precision), which is typically of order 10^{-8} or 10^{-16} in single or double precision computer arithmetic (Higham, 2002).

In this paper the PCD is applied in variance matrices, or in correlation matrices also, which should be positive definite. In case of $\mathbf{A}$ being a variance matrix of a random vector, the PCD has a reasonable interpretation. The first pivoting element indicates the random vector element with the largest variance, the second pivoting indicates the random vector element whose the variance conditioned on the first pivoted element is the largest, the third

pivoting indicates the random vector element whose the variance conditioned on the first two pivoted elements is the largest, and so on.

2.2.1 Positive Definite Ill-conditioned Matrices

When the symmetric positive definite matrix $\mathbf{A}$ is nearly singular, it is said to be *ill-conditioned*. A measure for ill-conditioning is given by the ratio of the largest to smallest eigenvalues of $\mathbf{A}$ called C. The matrix $\mathbf{A}$ is said to be singular if the ratio is infinite, and ill-conditioned if it is 'too large', where 'too large' means roughly $log(C)$ bigger than the precision of matrix entries, Golub and Loan (1996).

A positive definite ill-conditioned matrix, $\mathbf{A}$, has full rank, n, so the Pivoting Cholesky decomposition reduces to a simple Cholesky decomposition of a permutation of $\mathbf{A}$:

$$\mathbf{P^T A P} = \mathbf{R}^T\mathbf{R}, \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix}, \tag{3}$$

where the permutation $\mathbf{P}$ is given by the PCD algorithm, and for any $m \leq n$, $\mathbf{R}_{11} \in \Re^{n\text{-}m \times n\text{-}m}$ and $\mathbf{R}_{22} \in \Re^{m\times m}$ are upper triangular matrices, and $\mathbf{R}_{12} \in \Re^{n\text{-}m\times m}$. However, due to the ill-conditioning combined with numerical errors in the calculation, the stopping criterion of the PCD (2) can wrongly estimate the rank of $\mathbf{A}$ as $m < n$. Hence, the matrix $\mathbf{A}$ is treated as a singular matrix and the matrix $\mathbf{R}_{22}$ is set as zero matrix. But this approximation for $\mathbf{R}_{22}$ implies a false value for the determinant of $\mathbf{A}$, zero, and the inverse of $\mathbf{A}$ cannot be found. If the stopping criterion is not considered than the matrix $\mathbf{R}_{22}$ is inaccurately computed.

It is necessary to apply a small modification to the PCD for positive definite ill-conditioned matrix. The matrix $\mathbf{R}_{22}$ should be approximate by any upper triangular matrix, $\widehat{\mathbf{R}}_{22}$, with small values such as $\mathbf{R}_{11}(n,n) \geq \mathbf{R}_{22}(1,1) \geq \cdots \geq \mathbf{R}_{2,2}(m,m) \geq 0$, instead of setting it to a zero matrix. This approximation makes the determinat of $\mathbf{A}$ small and then it is possible to find an approximation for $\mathbf{A}^{-1}$. The approximate Pivoting Cholesky decomposition for a positive definite ill-conditioned matrix $\mathbf{A}$ is

$$\mathbf{P^T A P} \approx \mathbf{R}^T\mathbf{R}, \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \widehat{\mathbf{R}}_{22} \end{bmatrix}, \tag{4}$$

where a possible approximation for matrix $\mathbf{R}_{22}$ is given by $\widehat{\mathbf{R}}_{22}(1,1) = \mathbf{R}_{11}(m,m)/(m+1)$, and $\widehat{\mathbf{R}}_{22}(i,i) = \widehat{\mathbf{R}}_{22}(i\text{-}1, i\text{-}1)/(m+i)$ for $i = 2,\ldots,n-m$. The Algorithm 3 describes the procedure of how to decompose a positive definite ill-conditioned matrix $\mathbf{A}$ and find the permutation matrix $\mathbf{P}$.

Algorithm 3 This algorithm computes the approximate Pivoting Cholesky decomposition $\mathbf{P^T A P} \approx \mathbf{R}^T\mathbf{R}$ of a symmetric positive definite ill-conditioned matrix $\mathbf{A} \in \Re^{n\times n}$. The nonzero elements of the permutation matrix $\mathbf{P}$ are given by $\mathbf{P}(piv(k), k) = 1$, $k = 1, \ldots, n$

$\mathbf{R} = \mathbf{0}$ {define a $n \times n$ empty matrix}
$piv = 1 : n$
for $k = 1$ to n **do**
  $q = \{i : \mathbf{A}(i,i) = \max(diag(\mathbf{A}(k:n, k:n)))\} + k - 1$ {Finding the pivot}
  if Stopping criterion **then**
    $\mathbf{R}(j,j) = \mathbf{R}(j-1, j-1)/j, \quad \forall j = \{k, \ldots, n\}$
    stop {rank of $\mathbf{A}$ is $(k-1)$}
  end if
  $\mathbf{A}(:,k) \leftrightarrows \mathbf{A}(:,q)$ {Swap columns}
  $\mathbf{R}(:,k) \leftrightarrows \mathbf{R}(:,q)$ {Swap columns}
  $\mathbf{A}(k,:) \leftrightarrows \mathbf{A}(q,:)$ {Swap rows}
  $piv(k) \leftrightarrows piv(q)$ {Swap pivoting position}
  $\mathbf{R}(k,k) = \sqrt{\mathbf{A}(k,k)}$
  $\mathbf{R}(k, k+1:n) = \mathbf{R}(k,k)^{-1}\mathbf{A}(k, k+1:n)$
  $\mathbf{A}(k+1:n, k+1:n) = \mathbf{A}(k+1:n, k+1:n) - \mathbf{R}(k, k+1:n)^T\mathbf{R}(k, k+1:n)$
end for

In the next section, the Pivoting Cholesky decomposition is applied to Bayesian analysis of computer code output, where it is used as a tool to decompose variance matrices necessary for either estimation or prediction problems. Additionally, based on the PCD, validation diagnostics for Gaussian process emulators are built.

3 Statistical Analysis in Computer Models

3.1 Gaussian Process Emulator

In this section the technical details of the Gaussian process emulator are presented. Let $\eta(\cdot)$ be a representation of a computational model with input variables represented by the vector $\mathbf{x} = (x_1, \ldots, x_p) \in \chi_1 \times \cdots \times \chi_p = \chi \subset \Re^p$ and output variable represented by $y \in \Re$.

Although the computer code can be possibly known, its complexity allows the simulator, $\eta(\cdot)$, to be considered an unknown function. Then a stochastic representation of our judge-

ments can be done. The most commonly used emulator is the Gaussian process emulator, where our beliefs about the simulator are described using a Gaussian Process.

Definition 1 (Gaussian Process) *Gaussian process is a stochastic process* $\{\eta(x)\}$, $x \in \chi \subset \Re^p$ *such that every finite linear combination of the* $\eta(\cdot)$ *is normally distributed. Notation. If* $\eta(\cdot)$ *is a Gaussian process with mean function* $m(\cdot)$ *and covariance function* $V(\cdot,\cdot)$*, then we write* $\eta(\cdot) \sim GP(m(\cdot), V(\cdot,\cdot))$.

Formally, a Gaussian process emulator for $\eta(\cdot)$ can be defined by a Gaussian process with mean $m_0(\cdot)$ and covariance function $V_0(\cdot,\cdot)$. Using an hierarchical formulation

$$\eta(\cdot)|\beta, \sigma^2, \psi \sim GP\left(m_0(\cdot), V_0(\cdot,\cdot)\right), \tag{5}$$

where

$$\begin{aligned} m_0(x) &= h(x)^T\beta, \\ V_0(x,x') &= \sigma^2 C(x,x';\psi), \end{aligned}$$

$h(\cdot) : \chi \subset \Re^p \longmapsto \Re^q$ is a known function of the inputs, where q can be different from the input space dimension p, β is an unknown vector of coefficients, σ^2 is an unknown scale parameter, $C(\cdot,\cdot;\psi)$ is a known correlation function with the unknown vector of correlation parameters $\psi = [\psi_1, \psi_2, \ldots, \psi_p]$.

Suppose $\mathbf{y}^T = [\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)]$ contains n realizations of the simulator at design points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ on the parameter space χ; these data are called the training dataset. The design points are chosen according to a design for computer models, Santner et al. (2003). According to (5) the distribution of the outputs is multivariate normal as follows:

$$\mathbf{y}|\beta, \sigma^2, \psi \sim N_n\left(H\beta, \sigma^2\mathbf{A}\right), \tag{6}$$

where

$$\begin{aligned} H &= [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]^T, \\ \mathbf{A}_{i,j} &= C(\mathbf{x}_i, \mathbf{x}_j; \psi). \end{aligned}$$

Using standard techniques for conditioning in multivariate normal distributions, it can be shown that

$$\eta(\cdot)|\beta, \sigma^2, \psi, \mathbf{y} \sim GP\left(m_0^*(\cdot), V_0^*(\cdot,\cdot)\right), \tag{7}$$

where

$$\begin{aligned} m_0^*(x) &= h(x)^T\beta + t(x)^T\mathbf{A}^{-1}(\mathbf{y} - H\beta) \\ V_0^*(x,x') &= \sigma^2\left[C(x,x';\psi) - t(x)^T\mathbf{A}^{-1}t(x)\right] \end{aligned}$$

where $t(x) = (C(x,\mathbf{x}_1;\psi),\ldots,C(x,\mathbf{x}_n;\psi))^T$.

Using a weak prior for (β,σ^2), $p(\beta,\sigma^2) \propto \sigma^{-2}$, combining with (6) using Bayes Theorem, the posterior for (β,σ^2) is a Normal Inverse Gamma distribution, whose marginals are:

$$\beta|\mathbf{y},\sigma^2,\psi \sim N\left(\widehat{\beta},\sigma^2\left(H^T\mathbf{A}^{-1}H\right)^{-1}\right)$$

where $\widehat{\beta} = \left(H^T\mathbf{A}^{-1}H\right)^{-1}H^T\mathbf{A}^{-1}\mathbf{y}$, and

$$\sigma^2|\mathbf{y},\psi \sim \mathbf{y}^T\left(A^{-1} - \mathbf{A}^{-1}H\left(H^TA^{-1}H\right)^{-1}H^T\mathbf{A}^{-1}\right)\mathbf{y}\chi^2_{n-q}$$

Integrating (β,σ^2) out from the product of $\eta(\cdot)|\beta,\sigma^2,\psi,\mathbf{y}$ and $\beta|\mathbf{y},\sigma^2,\psi$ it can be shown that

$$\eta(\cdot)|\mathbf{y},\psi \sim \text{Student Process}\,(n-q,m_1(\cdot),V_1(\cdot,\cdot)) \tag{8}$$

where

$$m_1(x) = h(x)^T\widehat{\beta} + t(x)^T\mathbf{A}^{-1}(\mathbf{y} - H\widehat{\beta}), \tag{9}$$

$$\begin{aligned} V_1(x,x') &= \widehat{\sigma^2}\left[C(x,x';\psi) - t(x)^T\mathbf{A}^{-1}t(x)\right. \\ &+ \left.\left(h(x) - t(x)^T\mathbf{A}^{-1}H\right)(H^T\mathbf{A}^{-1}H)^{-1}\left(h(x') - t(x')^T\mathbf{A}^{-1}H\right)^T\right] \end{aligned} \tag{10}$$

$$\widehat{\sigma^2} = \frac{\mathbf{y}^T\left(\mathbf{A}^{-1} - \mathbf{A}^{-1}H\left(H^T\mathbf{A}^{-1}H\right)^{-1}H^T\mathbf{A}^{-1}\right)\mathbf{y}}{n-q-2}.$$

However, the correlation length parameters ψ are unknown. Setting a prior for this hyperparameter correlation vector as $p(\psi)$, it can be shown that

$$\begin{aligned} p(\psi|\mathbf{y}) &\propto p(\psi)\int\int p(y|\beta,\sigma^2,\psi)p(\beta,\sigma^2)d\beta d\sigma^2 \\ &\propto p(\psi)|\mathbf{A}|^{-\frac{1}{2}}|H^T\mathbf{A}^{-1}H|^{-\frac{1}{2}}(\hat{\sigma}^2)^{-\frac{n-q}{2}} \end{aligned} \tag{11}$$

where $\mathbf{A}$ and $\hat{\sigma}^2$ are functions of ψ, and an estimator for ψ can be, for example, the mode of $p(\psi|\mathbf{y})$, obtained by an optimization method such as Nelder-Mead, (Nelder and Mead, 1965). Then, the new emulator is the same as (8) with the estimated values for $\mathbf{A}$, $\widehat{\beta}$, and $\widehat{\sigma^2}$ using the estimated value of ψ.

Notice that the estimating process of ψ demands several evaluations of the determinant and inverse of the matrix $\mathbf{A}$, which is a function of ψ. So, an efficient method is necessary to calculate these quantities. See Gentle (1998) for estimation methods and decomposition methods applied in statistics. However, some combinations of ψ values and some particular training data can lead to an ill-conditioned matrix $\mathbf{A}$ implying values too small for $|\mathbf{A}|$ and too big for $\hat{\sigma}^2$, which can return inaccurate evaluations of (11) invalidating the estimating process of ψ.

The Pivoting Cholesky decomposition for ill-conditioned matrices not only provides an aproximate decomposition allowing approximations for determinants and inverses, but also identifies which elements contribute most to the near singularity.

3.2 Validation of Computer Models

It is a fact that learning about a particular process from computer experiments is cheaper than learning from real-world experiments. However, before using a simulator to investigate a real system, it should be validated. For computer science and engineering, there are some guides for verification and validation of computer models (USDoD (1996), AIAA (1998), FDA (2002)). The validation process of computer models is the process of checking if the computer model satisfies a certain criterion, that generally involves the comparison of computer model outputs to physical observations of the real process.

Some authors have presented statistical procedures for validating simulators using experimental errors where they compare real observations against the simulator outputs. Hills and Trucano (1999, 2001) use a χ^2 test to compare the simulator output with the real process output. Oberkampf and Trucano (2000) present a qualitative validation procedure applied to fluid dynamics. However, a proper statistical validation process should be able to compare the whole uncertainty of a statistical process against real outputs of this process. And exploring this idea a procedure for validating an emulator is built.

If an emulator can imitate a particular simulator in the whole input space, then this emulator can be considered valid. The statistical process of checking if a Gaussian process emulator built from a training dataset can represent a simulator, including uncertainty associated on the emulator, will be called Emulator Validation. A dataset used to validate the emulator is called the validation dataset. The validation dataset is used to validate an emulator built from some training data. The inputs of the validation dataset should cover the

region of the input space where the emulator is going to used to make predictions. Santner et al. (2003) present some methods for design of computer experiments, but design for the validation dataset is an unexplored area. It is not considered here.

The simplest way to do this comparison is to check how many times the simulator output is inside a 95% credibility interval from the emulator (8); it is expected that 95% of the times the simulator output value will lie in its interval. However, this comparison does not consider the correlation among the outputs. So, validation diagnostics can be built in order to allow this comparison.

Suppose that the GP emulator (8) is built with the training data, $\{(y_1, \mathbf{x}_1), \ldots, (y_n, \mathbf{x}_n)\}$, and it will be validated with the validation data $\{(y_1^*, \mathbf{x}_1^*), \ldots, (y_m^*, \mathbf{x}_m^*)\}$. Hills and Trucano (1999, 2001) use a χ^2 test to compare the simulator output with the real process, and the same idea can be used to compare emulator predictions with the simulator outputs under the same inputs. However, the correlations among the predictions are still ignored, and the natural extension of the χ^2 statistics which can allow for the correlations is the Mahalanobis distance. The Mahalanobis distance between the emulator and the simulator outputs at the validation inputs set is given by

$$MD(\mathbf{X}^*)|\mathbf{y} = d(\mathbf{X}^*)^T \left(V_1(\mathbf{X}^*, \mathbf{X}^*)\right)^{-1} d(\mathbf{X}^*) \tag{12}$$

where $\mathbf{X}^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*)$ are the validation inputs, $d(\mathbf{X}^*) = \eta(\mathbf{X}^*) - m_1(\mathbf{X}^*)$, $m_1(\mathbf{X}^*)$ is the predictive mean vector with elements (9) and $V_1(\mathbf{X}^* t, \mathbf{X}^*)$ is the predictive variance matrix with elements (10).

Given the training data, the emulator follows a Student process (8). So, the uncertainty about the simulator evaluated on the validation inputs is represented by a multivariate Student-t distribution with $n - q$ degrees of freedom, and mean vector and scale matrix given by $m_1(\mathbf{X}^*)$ and $V_1(\mathbf{X}^*, \mathbf{X}^*)$, respectively, i.e.

$$\eta((\mathbf{X}^*) \sim t_{n-q}\left(m_1(\mathbf{X}^*), V_1(\mathbf{X}^*, \mathbf{X}^*)\right). \tag{13}$$

Then, the distribution of the Mahalanobis distance (12) conditional on the training data and evaluated on the validation inputs is a scaled F-Snedecor with m and $n - q$ degrees of freedom, (Box and Tiao, 1973, Eq. 2.7.21), i.e.

$$MD(\mathbf{X}^*)|\mathbf{y} \sim mF_{m,n-q}. \tag{14}$$

If the GP emulator can really imitate the simulator, then it is expected that the observed Mahalanobis distance should be close to its mean, $E[MD(\mathbf{X}^*)|\mathbf{y}] = m\frac{n-q}{n-q-2}$ or mode,

$mode[MD(\mathbf{X}^*)|\mathbf{y}] = \frac{(n-q)(m-2)}{(n-q+2)}$. And associated uncertainty can be expressed by the variance $V[MD(\mathbf{X}^*)|\mathbf{y}] = 2m\frac{(n-q)^2(m+n-q-2)}{(n-q-2)^2(n-q-4)}$.

In order to calculate the Mahalanobis distance it is necessary to invert the predictive variance matrix (10). The solving methods involve matrix decomposition. The PCD, Algorithm 3, applied on the predictive variance matrix provides an intuitive interpretation. The first pivot is the validation data point with the largest predictive variance, the second pivot is the point with the largest predictive variance given the first pivot, the third pivot is the largest predictive variance given the first two pivots, and so on. Additionally, when the predictive variance matrix is ill-conditioned, the random variables which do not satisfy the stopping criterion (2) can be identified by the algorithm and checked to explore a possible reason for this behaviour. For example, some elements can be so highly correlated with the other elements that they bring very little new information for the predictive variance structure.

Standardized validation errors can be built based on the Mahalanobis distance (12), when the predictive variance matrix can be decomposed into $V_1(\mathbf{X}^*, \mathbf{X}^*) = \mathbf{G}\mathbf{G}^T$.

$$MD(\mathbf{X}^*)|\mathbf{y} = d(\mathbf{X}^*)^T \left(V_1(\mathbf{X}^*, \mathbf{X}^*)\right)^{-1} d(\mathbf{X}^*) = VE(\mathbf{X}^*)^T VE(\mathbf{X}^*) \tag{15}$$

where $VE(\mathbf{X}^*) = \mathbf{G}^{-1}d(\mathbf{X}^*)$ is called the validation error vector. There are different other methods to decompose the predictive variance in addition to the PCD, such as the eigen decomposition, Gentle (1998). The eigen decomposition gives $\mathbf{G} = \mathbf{E}^T\Lambda^{-1/2}$, where Λ is a diagonal matrix with the eigenvalues of $V_1(\mathbf{X}^*, \mathbf{X}^*)$ and $\mathbf{E}$ is a matrix with the correspondent eigenvectors. But, the PCD is used to build the validation errors due to its stability, advantages to handle numerical ill-conditioned matrices, and simplicity of interpretation.

The distribution of the validation errors, $VE(\mathbf{X}^*)$ derived from (8) is a multivariate standard Student-t with $n - q$ degrees of freedom, as follows

$$VE(\mathbf{X}^*) \sim t_{n-q}\left(\mathbf{0}, \mathbf{I}_m\right). \tag{16}$$

3.2.1 Possible validation diagnostics

In this section, some possible diagnostics for validating an emulator are presented. In order to illustrate each proposed diagnostic, a Gaussian process emulator is built to imitate a simple mathematical function. The mathematical form of the toy example is given by

$$\eta(x) = sin(2x) + (x/2)^2, \quad x \in (-5, 5). \tag{17}$$

The simulator (17) was run for 10 training data, selected to cover the whole input space. The training data are presented in Table 1, and the function with the training data are illustrated in Figure 1.

x	-4	-4.5	-2.67	-1.33	0	1.33	2	2.67	4	4.5
$y = \eta(x)$	4.99	4.65	2.59	-0.01	0	0.90	0.24	0.96	3.01	5.47

Table 1: The training dataset

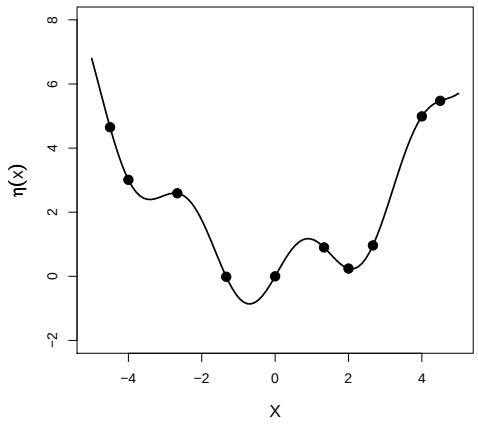


Figure 1: The toy simulator and the observed values (the black dots).

The training data were used to build the GP emulator, (8), where the functions $h(\cdot)$ and $C(\cdot, \cdot; \psi)$ defined in (5) are respectively $h(x) = (1, x)$ and $C(x, x'; \psi) = \exp\{-((x - x')/\psi)^2\}$. The estimated correlation length parameter using the Nelder-Mead optimization method, Nelder and Mead (1965), is $\hat{\psi} = 0.048$, and the estimated variance is $\widehat{\sigma}^2 = 18.45$.

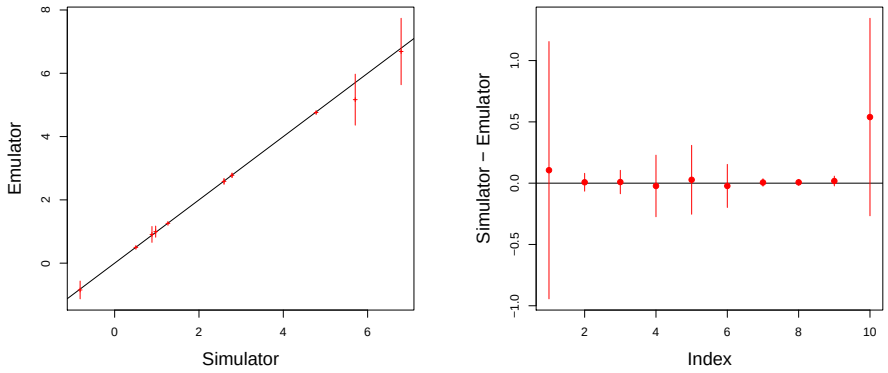
In order to validate the GP emulator (8) , a 10 element validation dataset was selected, as shown in Table 2.

x^*	-5	-3.89	-2.78	-1.67	-0.56	0.56	1.67	2.78	3.89	5
$y^* = \eta(x^*)$	6.79	2.78	2.59	0.89	-0.82	0.97	0.5	1.26	4.78	5.71

Table 2: The training dataset

The simplest diagnostic for check if the GP emulator (8) can represent the simulator (17) is to plot the outputs of the simulator against the predictive estimates from the emulator with an uncertainty measure, e.g. a credibility interval. If the emulator is a good approximation

for the emulator it is expected that the points follow a 45-degree line behaviour. One variation of this diagnostic is to plot the difference for the same input between the simulator output and the emulator prediction. In this case the expected behaviour for the points is a zero horizontal line. The Figure 2 illustrates these diagnostic plots for the toy simulator, and according to these diagnostics, the emulator seems a reasonable representation of the simulator.



(a) (b)

Figure 2: The marginal validation diagnostics: (a) prediction versus validation output; (b) Prediction error.

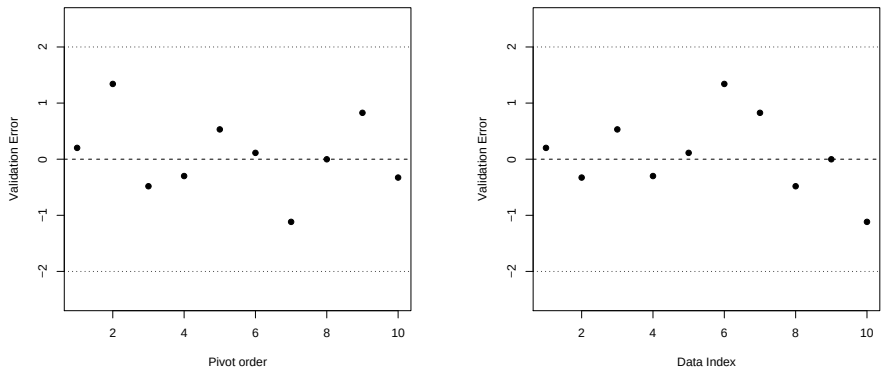
Additionally, a χ^2 statistic can also be presented ($\chi^2 = 3.287$), but neither this statistic nor the previous graphical diagnostics consider the correlation structure between the predicted elements. Table 3 presents the observed Mahalanobis distance, and some statistics of the predictive distribution of it which assumes that the emulator is a good representation of the emulator. The observed Mahalanobis distance is smaller but not so far from its expected value, small Mahalanobis distance can indicate large predictive variability. So apparently, the GP emulator with the training data is a good representation of the simulator.

Obs. M.D.	Mode	Mean	Std. dev.
4.495	6.400	13.333	11.926

Table 3: The observed Mahalanobis distance (M.D.) and some statistics of the predictive distribution of Mahalanobis distance.

Some graphical diagnostics can be built using the validation errors (15). Figure 3 (a) presents the observed validation errors against the pivots, i.e. the first element is the one

with the largest variance, the second element is the one with the largest variance conditioned on the first element, the third is the largest variance conditioned on the first and the second elements, and so on. It is expected that the validation errors are around zero and have most of the the points between -2 and 2. In order to identify which elements are responsible for each validation error the data can be reordered and then plotted as in Figure 3 (b). Notice that from the PCD it is possible to know exactly which element corresponds to each validation error.



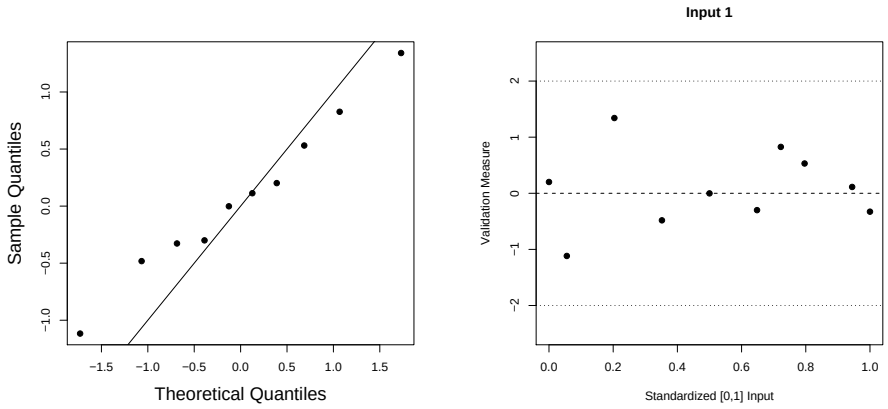
(a) (b)

Figure 3: Validation diagnostics: Observed validation errors (a) by the order of the PCD; (b) by the order of the observed data.

Additionally, the predictive distribution of the validation errors is known (16). Therefore, a QQ-plot can be drawn, as in Figure 4 (a). Where the slope of diagonal line is less than that of the data, the implication is that the predictive variances are larger than necessary. Also the validation errors can be plotted against the validation inputs in order to identify any pattern presented on the experimental design if there is any, the example is presented in Figure 4 (b).

According to Figure 2 the GP emulator that was built with the 10 training data presents reasonable estimates for the validation data. And the validation diagnostics, Table 3 and Figures 3 and 4, indicate that the built GP emulator is a valid approximation for the simulator.

Figure 5 confirms this, where it was predicted the output of the computer model and its uncertainty for 100-input equally spaced data on the interval (-5,5), where the predictive estimates have a similar behaviour as the true function and the uncertainty associated on the predictions is relatively low.



(a) (b)

Figure 4: Validation Diagnostics: (a) Quantile-quantile plot of the validation errors; (b) relationship between the standardized [0,1] input variable and the validation errors.

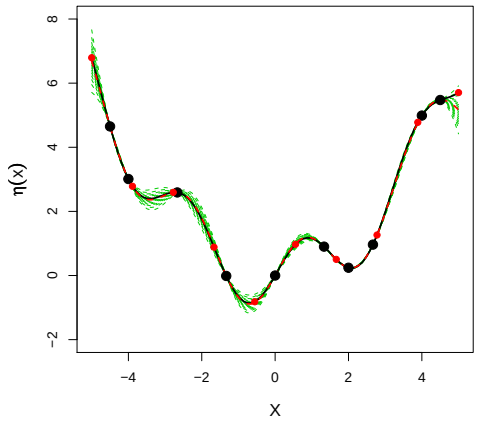


Figure 5: The GP emulator output for the toy simulator: True function (black line), predictive posterior mean (red dotted line), quantiles 5%, 10%,...,95% of the predicive distribution (green dotted lines), the training data (black dots) and the validation data (red dots).

4 Examples

4.1 Two-input toy model

A known mathematical function with 2 inputs is used as simulator in this Section. A Gaussian process emulator will be built with some training data, and in order to validate the emulator, the previous presented diagnostics using validation data will be used. The toy simulator, $\eta(\cdot,\cdot)$, is the following function

$$y = \eta(x_1, x_2) = \left(1 - e^{-\frac{1}{2x_2}}\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right), \quad x_i \in (0,1), i = 1, 2. \tag{18}$$

The training data is composed of 20 points selected using Latin Hypercube sampling, Santner et al. (2003). The GP emulator is given by (5) where the functions $h(\cdot) = (1, x)$, $C(\cdot,\cdot;\psi) = \exp\{-(x - x')^T diag(\psi_1^{-\frac{1}{2}}, \psi_2^{-\frac{1}{2}})(x - x')\}$, and the correlation length parameters (ψ_1, ψ_2) are unknown. The validation data is composed of 25 points also selected using Latin Hypercube sampling. The training and the validation data are presented in Figure 6.

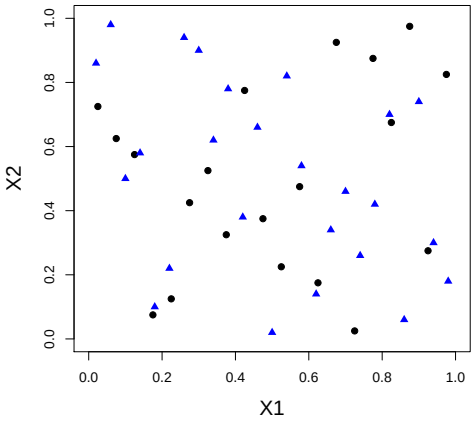
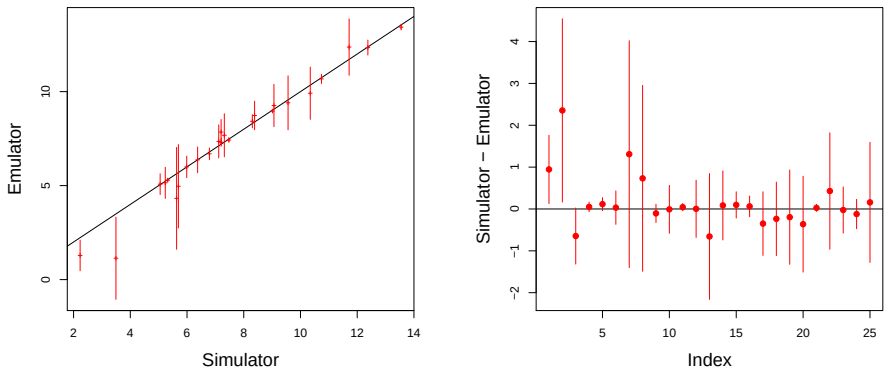


Figure 6: Training data (●) and validation data (▲) from Latin Hypercube sampling.

The estimated correlation length parameters using the Nelder-Mead optimization method applied to function (11) are $(\hat{\psi}_1, \hat{\psi}_2) = (0.0042, 0.0397)$, and the estimated variance is $\widehat{\sigma}^2 = 3.3327$. The GP emulator (8) is built assuming that the estimated correlation length parameters are true values of them. The emulator mean (9) is used to predicte the computer model for any given set of inputs. Figure 7 (a) and (b) indicate that the emulator

preditictions for the validation data are quite accurate for some inputs, but for others the predictions are far from the true simulator value.



(a) (b)

Figure 7: The marginal validation diagnostics: (a) prediction versus validation output; (b) Prediction error.

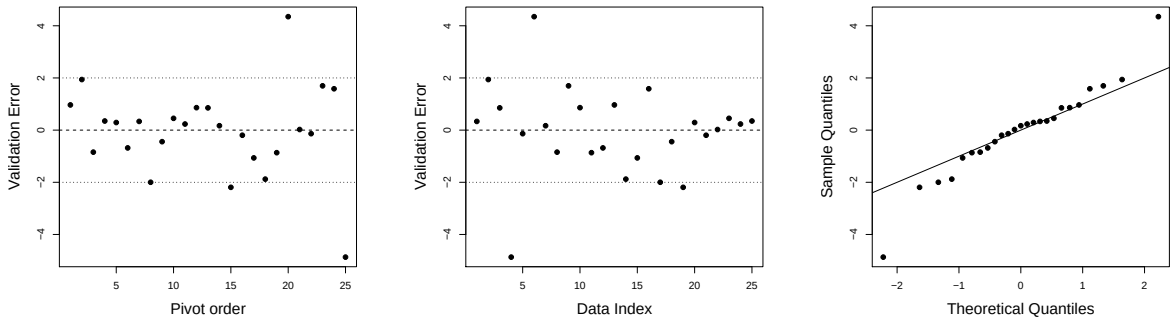
Table 4 presents the observed Mahalanobis distance, and some statistics of the predictive distribution of it. The observed Mahalanobis distance is bigger than its expected value, this can indicate small predictive variability. And apparently, the built GP emulator with the training data is not a good representation of the simulator. But the validation error decomposed from the Mahalanobis distance can help to identify why the observed Mahalanobis distance assumes this value.

Obs. M.D.	Mode	Mean	Std. dev.
70.4256	20.5789	28.3333	14.0573

Table 4: The observed Mahalanobis distance (M.D.) and some statistics of the predictive distribution of Mahalanobis distance.

Figure 8 (a) and (b) present the validation errors (15) ordered by the pivots from the PCD and by the validation data ordering, respectively. The 4th (the 25th pivot) and the 6th (the 20th pivot) validation data have presented the biggest validation errors, -4.868, and 4.349. Only these two validation errors represent 60.51% of the observed Mahalanobis distance. The big validation errors may be caused by either a small predictive uncertainty or by a high correlation among the previous pivots. For example, the predictive emulator correlation between the 6th and the 5th (the 22nd pivot) validation data is 0.949, and

the correlation between the 4th and the 3rd (the 13rd pivot) validation data is -0.9322, in this last case they are highly negative correlated. These elements are highly correlated, then if they are not considered then the observed Mahalanobis distance reduces to 27.807 indicating the emulator as a good approximation for the simulator, because it is quite close to the recalculated expected value 26.067. In Figure 8 (c) the slope of the diagonal line is smaller than that of the data, but apart from the two biggest validation errors the diagonal line fits very well the data giving more information to believe that the emulator is a good aproximation for the simulator.



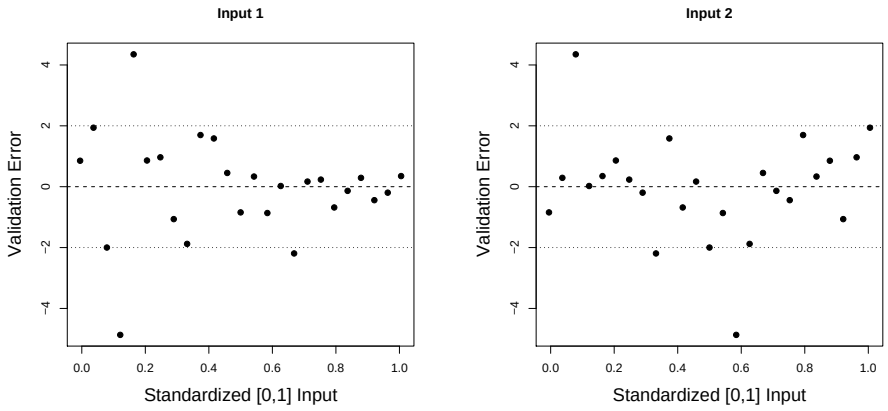
(a) (b) (c)

Figure 8: Validation diagnostics: Observed validation errors (a) by the order of the PCD; (b) by the order of the observed data; (c) QQ-plot of the validation errors.

The validation errors are plotted against the two validation inputs, Figure 9 (a) and (b). For the first inputs, the validation errors are smaller for large values of the input 1 (X_1), which can mean that that predictivy variability for this input region is large. For the second input, it is not clear a particular pattern for the validation errors.

The previous diagnostics incate that the emulator built with the training data can represent the emulator, and predictions from the emulator can be done for any input on the input space. Figure 10 (a) presents emulator preditions over the input space. Figure 10 (b) presents the associated variability, as expected the variability is higher on regions where the training data were not observed.

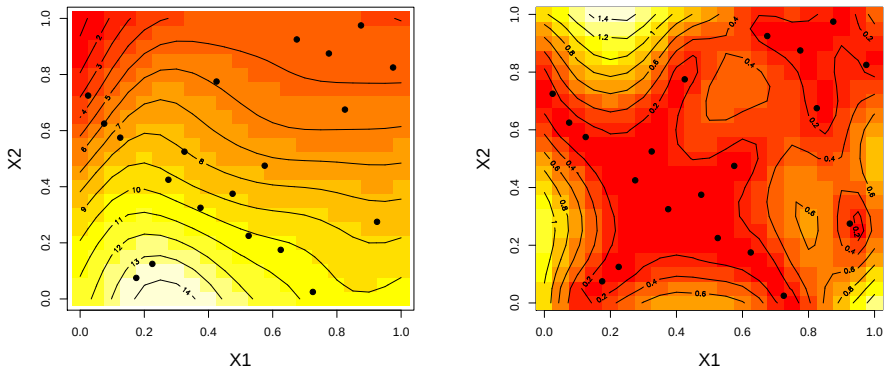
Since the simulator (18) is a simple mathematical function, its true values over the input space are presented on Figure 11 (a). As the validation diagnostics pointed out the GP emulator is a good representation of the simulator. The prediction errors, difference between the emulator prediction and the validation predictions, are presented on Figure 11 (b), where it is clear that for small values of input 1 the predicitve mean is not a good approximation.



(a) (b)

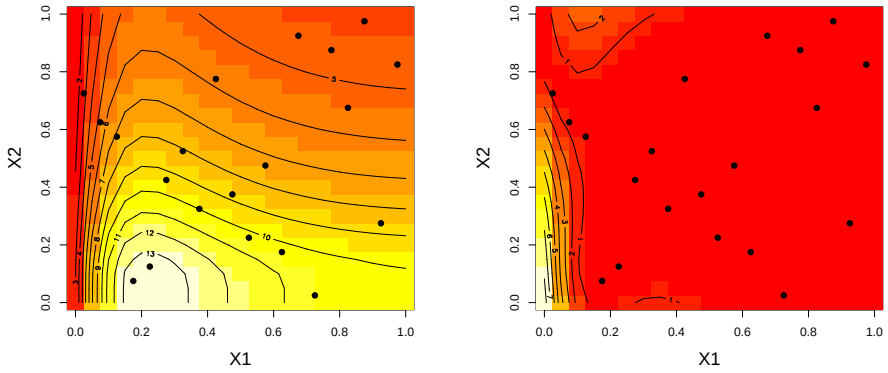
Figure 9: Relationship between the standardized [0,1] input variable and the validation errors: (a) Input 1 (X_1); (b) Input 2 (X_2)

Although there are very few training data for small values of input 1, the predictive variance over this region is high.



(a) (b)

Figure 10: The predictive mean (a), and the predictive standard deviation (b) obtained by the GP Emulator built with the training data identified by (•).



(a) (b)

Figure 11: Two dimentional toy simulator (a) and the predictive error for the GP Emulator (b).

4.2 Nilson Kuusk Model

In this section, a real dataset is used as an example of the validation diagnostics. The simulator was built based on the Nilson-Kuusk model, which is a reflectance model for a homogeneous plant canopy, Nilson and Kuusk (1989). The simulator has 5 inputs, the solar zenith angle with range $[0, 87]$, the leaf area index $[0, \infty]$, relative leaf size $[0, \infty]$, the Markov clumping parameter $[0.4, 1.5]$ and λ with range [550,850]. The real meaning of these inputs and the outputs it is not part of these work, but for more details about this problem see Nilson and Kuusk (1989) and Kuusk (1996). Here the inputs are referenced as input 1 to input 5.

The training data and the validation data used here are example datasets of the GEM-SA software (*http://ctcd.group.shef.ac.uk/gem.html*), and they contain 150 and 100 points, respectively. The GP emulator, (8), for the Nilson-Kuusk model was built using the training data, where the functions $h(\cdot)$ and $C(\cdot, \cdot; \psi)$ defined in (5) are respectively $h(x) = (1, x)$ and $C(x, x'; \psi) = \exp\{-(x - x')^T \Omega (x - x')\}$, $\Omega = diag(\psi_1^{-\frac{1}{2}}, \ldots, \psi_5^{-\frac{1}{2}})$.

The estimated correlation length parameters were estimated using the L-BFGS-B optimization method, Byrd et al. (1995), applied to the marginal distribution of ψ; the estimates are presented in Table 5, and the estimated variance is $\widehat{\sigma}^2 = 0.0072^2$.

$\hat{\psi}_1$	$\hat{\psi}_2$	$\hat{\psi}_3$	$\hat{\psi}_4$	$\hat{\psi}_5$
0.573	0.899	2.757	1.936	0.144

Table 5: Correlation length estimates.

The prediction errors of the GP emulator for the validation data are shown in Figure 12, where the 2-standard-deviation credible intervals are presented. It can be noticed that approximately 95% of the intervals should contain zero, which roughly indicates that the emulator is a good approximation for the emulator.

However, the observed Mahalanobis distance for the validation data is bigger than expected, which means that the GP emulator is not a good approximation for the Nilson-Kuusk model, Table 6. The difference between the χ^2 statistic and the observed Mahalanobis distance indicates that the correlation among the elements is significant, because the χ^2 statistic is a non-correlated case of the Mahalanobis distance.

Figure 13 (a) presents the validation errors (15) ordered by the PCD, where the k-th validation error depends on the $(k-1)$ pivoted elements previously calculated. Figure 13

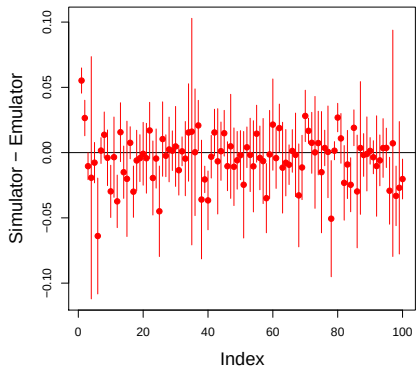


Figure 12: The validation errors and the marginal uncertainty.

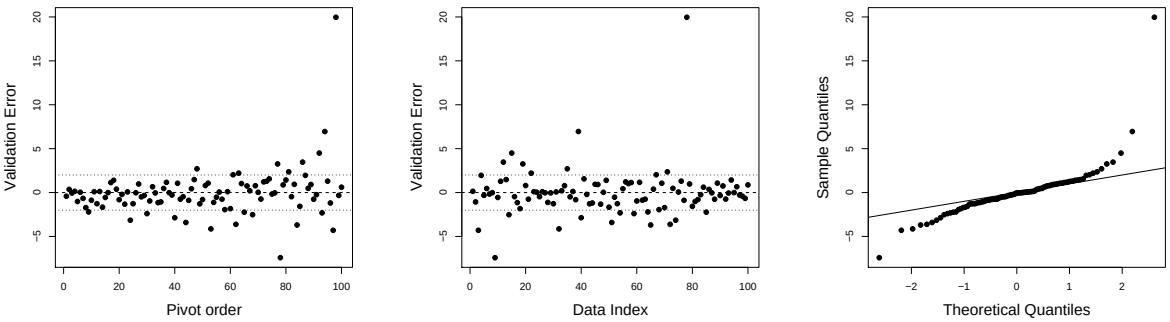
χ^2	Observed MD	Mode	Mean	s.d.
374.8484	752.8213	96.6776	101.4085	18.8553

Table 6: The χ^2 statistic of the validation errors, the observed Mahalanobis distance (M.D) and some statistics of its predictive distribution.

(b) presents the validation errors by the data observed order that helps to identify exactly which are the elements with large validation errors. And Figure 13 (c) shows the QQ-plot of the validation error with a heavy tail behaviour.

In order to find whether there is a particular subspace in the input space where the validation errors are larger, the validation errors are plotted against each input variable, Figure 14. It is clear that larger validation errors occur only for larger values of input 5.

For Figure 14, the value 0.5 of the standardised input 5 seems to be a change-point. Based on that, the training outputs are plotted against the training inputs, Figure 15. And the elements smaller than the change-point, 700 on the original scale, are identified where the change-point identified by the validation diagnostic splits the outputs into two distinct groups.



(a) (b) (c)

Figure 13: Validation diagnostics: Observed validation errors (a) by the order of the PCD; (b) by the order of the observed data; (c) QQ-plot of the validation errors.

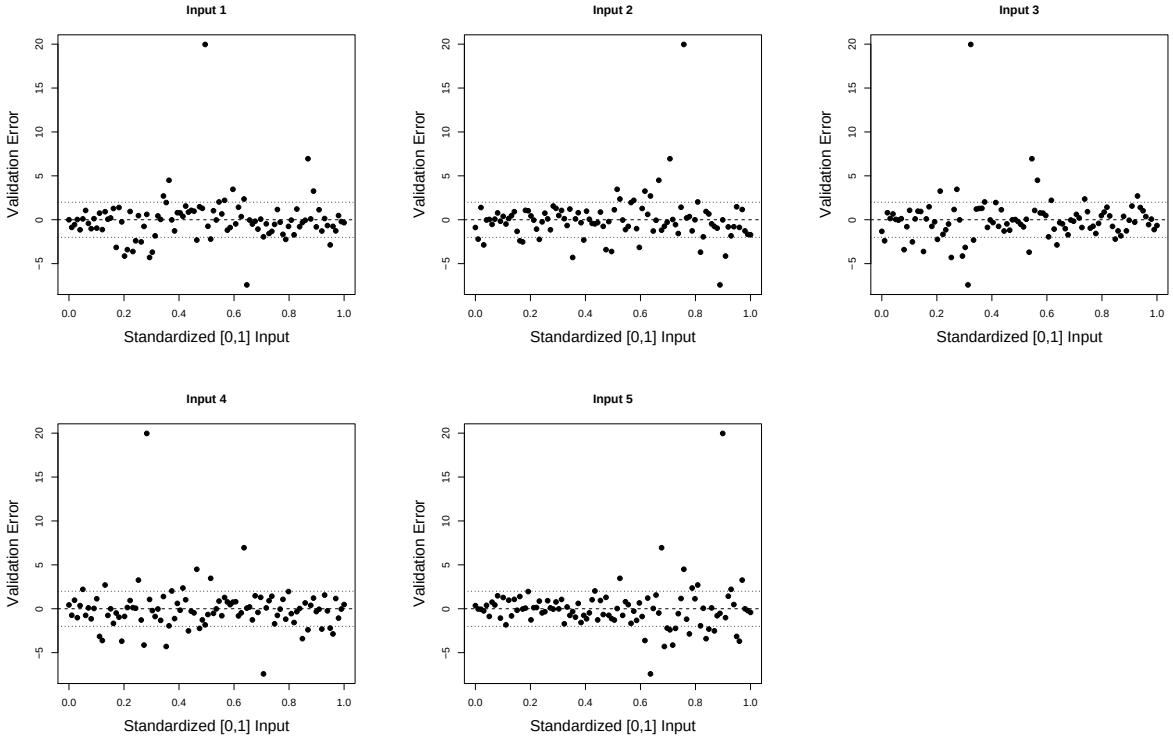


Figure 14: Relationship between the validation error and the standardized input variables.

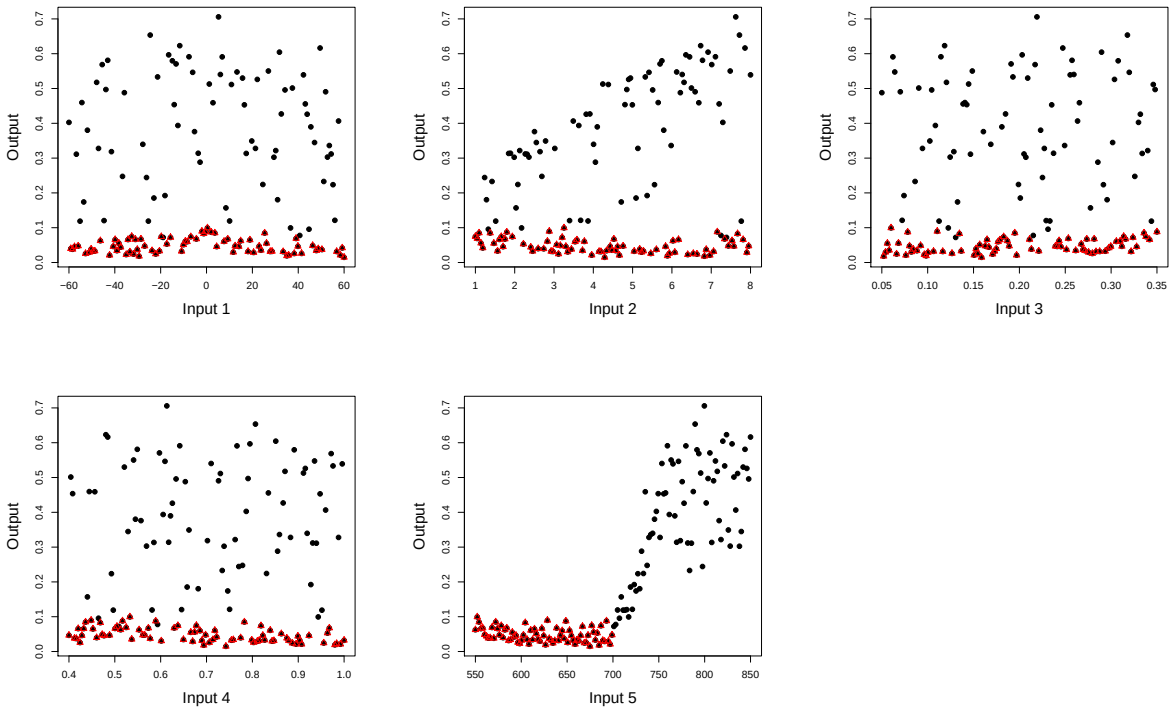


Figure 15: Relationship between inputs and the output of the training data. The red dots represent the values where the 5th input is smaller than 700 (the visual change-point).

5 Discussion

This work presents a modified pivoting Cholesky decomposition for ill-conditioned matrices, which can be useful when determinants and inversions of positive definite matrices are required. In particular, this decomposition is applied in the emulator setting, where it is used to build some diagnostics for validating a Gaussian process emulator. The validation errors (15) and the diagnostics based on them may identify when the emulator is not a good approximation for the simulator.

The same idea of the validation diagnostics for the emulators can be applied to validation of predictors (O'Hagan, 2006). The predictors use both experimental and simulator data to make predictions of the real process. Notice that the PCD is also needed because evaluations of determinants and inversions of positive matrices are required.

References

AIAA. *Guide for the verification and validation of computational fluid dynamics simulations.* American Institute of Aeronautics and Astronautics, AIAA-G-077-1998, 1998.

Commandant Benoît. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un systàme d'équations linéaires en nombre inférieur à celui des inconnues – application de la méthode à la résolution d'un systéme défini d'équations linéaires (procédé du commandant cholesky). *Bulletin Géodésique*, 2:67–77, 1924.

George E. P. Box and George C. Tiao. *Bayesian Inference in Statistical Analysis.* John Wiley and Sons, Inc., 1973.

C. Brezinski. The life and work of andré cholesky. *Numerical Algorithms*, 43:279–288, 2006.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ci You Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(6):1190–1208, 1995.

FDA. *General principles of software validation; Final guidance for Industry and FDA staff.* U.S. Food and Drug Administration, 2002.

L. Fox, H. D. Huskey, and J. H. Wilkinson. Notes on the solution of linear algebraic equations. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1:149–173, 1948.

James E. Gentle. *Numerical Linear Algebra for Applications in Statistics*. Springer, 1998.

Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

Sven Hammarling, Nicholas J. Higham, and Craig Lucas. Lapack-style codes for pivoted cholesky and qr updating. MIMS EPrint 2006.385, Manchester Institute for Mathematical Sciences, University of Manchester, Manchester, UK, jan 2007.

D. A. Harville. *Matrix algebra from a statistician's perspective*. Springer, 1997.

R. G. Haylock and A. O'Hagan. On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. Technical Report 94-18, Department of Mathematics, University of Nottihgham, 1994.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. ISBN 0-89871-521-0. Second Edition. First edition 1996.

Richard G. Hills and Timothy G. Trucano. Statistical validation of engineering and scientific models: A maximum likelihood based metric. Technical Report SAND2001-1783, Sandia National Laboratory, 2001.

Richard G. Hills and Timothy G. Trucano. Statistical validation of engineering and scientific models: Background. Technical Report SAND99-1256, Sandia National Laboratory, 1999.

A. Kuusk. A computer-efficient plant canopy reflectance model. *Computers and Geosciences*, 22:149–163, 1996.

J. A. Nelder and R. Mead. A simplex algorithm for function minimization. *Computer Journal*, 7:308–313, 1965.

Tiit Nilson and Andres Kuusk. A reflectance model for the homogeneous plant canopy and its inversion. *Remote Sensing of Environment*, 27(2):157–167, 1989.

Jeremy E. Oakley and Anthony O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784, 2002.

Jeremy E. Oakley and Anthony O'Hagan. Probabilistic sensitivity analysis of complex models: a bayesian approach. *Journal of the Royal Statistical Society B*, 66(3):751–769, 2004.

W. Oberkampf and T. Trucano. Validation methodology in computational fluid dynamics. Technical Report 2000-2549, American Institute of Aeronautics and Astronautics, 2000.

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

Anthony O'Hagan, Marc C. Kennedy, and Jeremy E. Oakley. Uncertainty analysis and other inference tools for computer codes. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 6*, pages 503–524. Oxford University, 1998.

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

T. J. Santner, Williams B., and Notz W. *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.

Olga Taussky and John Todd. Cholesky, toeplitz and the triangular factorization of symmetric matrices. *Numerical Algorithms*, 41:197–202, 2006.

A M Turing. Rounding-off errors in matrix processes. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1:287–308, 1948.

USDoD. *Verification, Validation, and Accreditation (VV&A) Recommended Practices Guide*. United States Department of Defense, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering, 1996.

W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris. Screening, predicting, and computer experiments. *Technometrics*, 34:15–25, 1992.