NEURAL COMPUTING
NCRG
RESEARCH GROUP

Neural Computing Research Group
Aston University
Birmingham B4 7ET
United Kingdom
Tel: +44 (0)121 333 4631
Fax: +44 (0)121 333 4586
http://www.ncrg.aston.ac.uk/

Emulation of dynamic computer models with multivariate output

Remi BARILLEC, Alexis BOUKOUVALAS, Dan CORNFORD

Unassigned Technical Report April 21, 2009

Abstract

This preliminary report describes work carried out as part of work package 1.2 of the MUCM research project. The report is split in two parts: the first part (Sections 1 and 2) summarises the state of the art in emulation of computer models, while the second presents some initial work on the emulation of dynamic models. In the first part, we describe the basics of emulation, introduce the notation and put together the key results for the emulation of models with single and multiple outputs, with or without the use of mean function. In the second part, we present preliminary results on the chaotic Lorenz 63 model. We look at emulation of a single time step, and repeated application of the emulator for sequential prediction. After some design considerations, the emulator is compared with the exact simulator on a number of runs to assess its performance. Several general issues related to emulating dynamic models are raised and discussed. Current work on the larger Lorenz 96 model (40 variables) is presented in the context of dimension reduction, with results to be provided in a follow-up report. The notation used in this report are summarised in appendix.

Contents

1 Single output emulation

In this section, we summarise the methods for the emulation of a univariate simulator $\eta : \mathbb{R}^n \to \mathbb{R}$. The simulator is run on a set of inputs $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \in \mathbb{R}^{n,N}$, yielding a set of target outputs $\mathbf{y} = \{y_1, ..., y_N\}$. The set of input/output pairs $(\mathbf{x}_i, y_i)_{i=1:N}$ is used to train an emulator $f : \mathbb{R}^n \to \mathbb{R}$ (i.e. a Gaussian Process) and is referred to as *training set*. Prediction consists in using the emulator to predict target outputs $\mathbf{y}_* = \{y_{*1}, ..., y_{*M}\}$ at a given set of input points $\mathbf{X}_* = \{\mathbf{x}_{*1}, ..., \mathbf{x}_{*M}\}$. The set of input/output pairs $(\mathbf{x}_{*i}, y_{*i})_{i=1:M}$ is referred to as *test set*.

1.1 Without mean function

We first consider the case of a zero-mean Gaussian Process. See for instance Bishop (2006); Rasmussen and Williams (2006). A covariance function k defines the correlations between inputs, so that the covariance between two inputs $\mathbf{x}$ and $\mathbf{x}'$ is given by the scalar $k(\mathbf{x}, \mathbf{x}')$. More generally, for two given sets of inputs $\mathbf{X}$ and $\mathbf{X}'$ of sizes N and N' respectively, k induces an $N \times N'$ covariance matrix $K = k(\mathbf{X}, \mathbf{X}')$. In particular, we denote $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$ the covariance matrix for the training set. The parameters of the covariance function (also called hyperparameters) are denoted θ.

1.1.1 Prediction

Prediction at a single point

For a single test point $\mathbf{x}_*$, the predicted output (as given by the emulator) has a

Gaussian distribution $\mathrm{p}(\mathbf{y}_*|\mathbf{x}_*, \mathbf{X}, \theta)$ with mean μ_* and covariance σ_*^2 given by:

$$\underset{1,1}{\mu_*} = \underset{1,N}{\mathbf{k}^{\mathrm{T}}}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,1}{\mathbf{y}} \tag{1}$$

$$\underset{1,1}{\sigma_*^2} = \underset{1,1}{\mathbf{k}_*} - \underset{1,N}{\mathbf{k}^{\mathrm{T}}}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,1}{\mathbf{k}}\ , \tag{2}$$

where $\mathbf{k} = k(\mathbf{X}, \mathbf{x}_*)$ and $\mathbf{k}_* = k(\mathbf{x}_*, \mathbf{x}_*)$. This is easily extended to the case of multiple test points:

Prediction at a set of points

$$\underset{M,1}{\mu_*} = \underset{M,N}{\mathbf{k}^{\mathrm{T}}}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,1}{\mathbf{y}} \tag{3}$$

$$\underset{M,M}{\mathbf{\Sigma}_*} = \underset{M,M}{\mathbf{k}_*} - \underset{M,N}{\mathbf{k}^{\mathrm{T}}}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,M}{\mathbf{k}}\ , \tag{4}$$

where $\mathbf{k} = k(\mathbf{X}, \mathbf{X}_*)$ and $\mathbf{k}_* = k(\mathbf{X}_*, \mathbf{X}_*)$. Note that it is common in the Machine Learning community to only compute the diagonal of the matrix $\mathbf{\Sigma}_*$.

1.1.2 Estimation of the hyperparameters

Marginal maximum likelihood

'Optimal' values for the hyperparameters can be estimated from the training data through maximisation of the marginal likelihood (or its logarithm, which is equivalent but simplifies the computation):

$$\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\ln|\mathbf{K}| - \frac{N}{2}\ln 2\pi. \tag{5}$$

Derivatives of the marginal maximum likelihood

The minimisation usually requires computation of the derivatives:

$$\frac{\partial}{\partial\theta_j}\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{Tr}(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}) \tag{6}$$

$$= \frac{1}{2}\mathrm{Tr}\left((\alpha\alpha^{\mathrm{T}} - \mathbf{K}^{-1})\frac{\partial\mathbf{K}}{\partial\theta_j}\right) \text{ with } \alpha = \mathbf{K}^{-1}\mathbf{y} \tag{7}$$

1.2 With mean function

Following Rasmussen and Williams (2006), the mean function can be represented by a set of q basis functions $\mathbf{h} = (h_1, ..., h_q)^T$ with regression coefficients β to be estimated from the data. This yields a Gaussian Process g defined as:

$$g(\mathbf{x}) = \underset{1,q}{\mathbf{h}(\mathbf{x})^{\mathrm{T}}}\ \underset{q,1}{\beta} + f(\mathbf{x}) \tag{8}$$

where $f(\mathbf{x})$ is a zero-mean Gaussian Process.

1.2.1 Prediction (informative prior)

Prediction with mean function (prior)

Given a prior on $\beta \sim \mathcal{N}(\mathbf{b}, \mathrm{B})$, the uncertainty in β can be integrated out, giving the predicted distribution for a test set $\mathbf{X}_*$:

$$\underset{M,1}{\mu'_*} = \underset{M,q}{H_*^{\mathrm{T}}}\ \underset{q,1}{\bar{\beta}} + \underset{M,N}{\mathbf{k}^{\mathrm{T}}}\ \underset{N,N}{\mathbf{K}^{-1}}(\ \underset{N,1}{\mathbf{y}} - \underset{N,q}{H^{\mathrm{T}}}\ \underset{q,1}{\bar{\beta}}\ ) \tag{9}$$

$$= \underset{M,1}{\mu_*} + R^{\mathrm{T}}\bar{\beta} \tag{10}$$

$$\underset{M,M}{\mathbf{\Sigma}'_*} = \underset{M,M}{\mathbf{\Sigma}_*} + R^{\mathrm{T}}(\mathrm{B}^{-1} + H\mathbf{K}^{-1}H^{\mathrm{T}})^{-1}R \tag{11}$$

where

$$\underset{q,N}{H} = \mathbf{h}(\mathbf{X}) \tag{12}$$

$$\underset{q,M}{H_*} = \mathbf{h}(\mathbf{X}_*) \tag{13}$$

$$\underset{q,M}{R} = \underset{q,M}{H_*} - \underset{q,N}{H}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,M}{\mathbf{k}}\ , \tag{14}$$

$$\underset{q,1}{\bar{\beta}} = (\underset{q,q}{\mathrm{B}^{-1}} + \underset{q,N}{H}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,q}{H^{\mathrm{T}}})^{-1}(\underset{q,N}{H}\ \underset{N,N}{\mathbf{K}^{-1}}\ \underset{N,1}{\mathbf{y}} + \underset{q,q}{\mathrm{B}^{-1}}\ \underset{q,1}{\mathbf{b}}). \tag{15}$$

1.2.2 Prediction (uninformative prior)

In the case of an uninformative prior, i.e. when $\mathrm{B}^{-1} \to 0$, the expressions simplify to:

Prediction with mean function (no prior)

$$\mu'_* = \mu_* + R^{\mathrm{T}}\bar{\beta} \tag{16}$$

$$\mathbf{\Sigma}'_* = \mathbf{\Sigma}_* + R^{\mathrm{T}}(H\mathbf{K}^{-1}H^{\mathrm{T}})^{-1}R \tag{17}$$

where

$$\bar{\beta} = (H\mathbf{K}^{-1}H^{\mathrm{T}})^{-1}H\mathbf{K}^{-1}\mathbf{y}. \tag{18}$$

1.2.3 Estimation of the hyperparameters (informative prior)

The marginal log-likelihood is given by:

MLL with mean function (prior)

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\mathbf{b},\mathrm{B}) = &-\frac{1}{2}(H^{\mathrm{T}}\mathbf{b}-\mathbf{y})^{\mathrm{T}}(\mathbf{K}+H^{\mathrm{T}}\mathrm{B}H)^{-1}(H^{\mathrm{T}}\mathbf{b}-\mathbf{y})\\ &-\frac{1}{2}\ln|\mathbf{K}+H^{\mathrm{T}}\mathrm{B}H| - \frac{N}{2}\ln 2\pi \qquad (19)\\ \propto &-\frac{1}{2}\,\mathrm{d}^{\mathrm{T}}G^{-1}\mathrm{d} - \frac{1}{2}\ln|G| \qquad (20)\end{aligned}$$

where we used $\mathrm{d} = H^{\mathrm{T}}\mathbf{b} - \mathbf{y}$ and $G = \mathbf{K} + H^{\mathrm{T}}\mathrm{B}H$.

The corresponding derivatives follow:

Derivatives of MLL with mean function (prior)

$$\frac{\partial}{\partial\theta_j}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\mathbf{b},\mathrm{B}) = \frac{1}{2}\mathrm{Tr}\left((\alpha\alpha^{\mathrm{T}} - G^{-1})\frac{\partial G}{\partial\theta_j}\right) \tag{21}$$

with $\alpha = G^{-1}\mathrm{d}$. It is worth observing that:

$$\frac{\partial G}{\partial\theta_j} = \frac{\partial\mathbf{K}}{\partial\theta_j}. \tag{22}$$

1.2.4 Estimation of the hyperparameters (uninformative prior)

In the case of an uninformative prior, the marginal log-likelihood becomes (Rasmussen and Williams, 2006):

MLL with mean function (no prior)

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = &-\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}|\\ &+\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{C}\mathbf{y} - \frac{1}{2}\log|\mathbf{A}| - \frac{n-m}{2}\log 2\pi,\end{aligned} \tag{23}$$

where:

$$m = \text{rank}(H), \tag{24}$$
$$\mathbf{A} = H\mathbf{K}^{-1}H^{\mathrm{T}}, \tag{25}$$
$$\mathbf{C} = \mathbf{K}^{-1}H^{\mathrm{T}}\mathbf{A}^{-1}H\mathbf{K}^{-1}. \tag{26}$$

The derivative of the marginal log-likelihood is given by:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{Tr}(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j})+ \\ \frac{1}{2}\mathbf{y}^{\mathrm{T}}\frac{\partial\mathbf{C}}{\partial\theta_j}\mathbf{y} - \frac{1}{2}\mathrm{Tr}(\mathbf{A}^{-1}\frac{\partial\mathbf{A}}{\partial\theta_j})\end{aligned} \tag{27}$$

with

$$\frac{\partial\mathbf{C}}{\partial\theta_j} = (-2\mathbf{K}^{-1}+\mathbf{C})\frac{\partial\mathbf{K}}{\partial\theta_j}\mathbf{C} \tag{28}$$
$$\frac{\partial\mathbf{A}}{\partial\theta_j} = -H\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\mathbf{K}^{-1}H^{\mathrm{T}} \tag{29}$$

2 Multi-output emulation

The aim of the method described in this section is the emulation of a mapping $\boldsymbol{\eta}$ from an n-dimension input space to an p-dimension output space. For each input $\mathbf{x} \in \mathbb{R}^n$, $\boldsymbol{\eta}$ gives an output $\mathbf{y} = \boldsymbol{\eta}(\mathbf{x}) \in \mathbb{R}^p$. Typically, the emulator is trained using a training set of N input/output pairs. The training inputs are stored as the columns of a design matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathbb{R}^{n,N}$ while the training outputs are stored in a similar matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N) \in \mathbb{R}^{p,N}$. It is convenient, for computation purposes, to vectorise the $p \times N$ output matrix into a $pN \times 1$ vector $\tilde{\mathbf{y}}$.

We consider here the case of a separable emulator, in which the covariance on the output is separated from the covariance on the input. A covariance function k is specified on the input space just as in the single output case. The covariance between outputs is specified by a fixed covariance matrix $K_f$indicating the correlation between output components (i.e. output dimensions), so that for two input points, the covariance of corresponding emulator's responses is given by: $\langle\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')\rangle = K_f \times k(\mathbf{x}, \mathbf{x}')$. Further assuming that the input covariance function is isotropic, we have: $\langle\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x})\rangle = K_f$.

2.1 Without mean function

2.1.1 Prediction

Single test point The emulator $\mathbf{f}$ is used to predict the output at a new input $\mathbf{x}_*$, giving a test input-output pair $\mathbf{y}_* = \mathbf{f}(\mathbf{x}_*)$. In the absence of noise on the output, the marginal distribution of the test output $\mathbf{y}_*$ is (Bonilla et al., 2007) $p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{Y}) = \mathcal{N}(\mu_*, \boldsymbol{\Sigma}_*)$, with:

Prediction at a single point

$$\underset{p,1}{\mu_*} = \underset{p,pN}{\mathbf{k}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,1}{\tilde{\mathbf{y}}} \tag{30}$$
$$\underset{p,p}{\boldsymbol{\Sigma}_*} = \underset{p,p}{\mathbf{k}_*} - \underset{p,pN}{\mathbf{k}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,p}{\mathbf{k}} \tag{31}$$

where $\mathbf{k}$ denotes the covariance between the test point and the training set, $\mathbf{K}$ denotes the training set's covariance and $\mathbf{k}_*$ is the covariance of the test point. We use the following shortcut notations for the various covariance matrices:

$$\underset{pN,pN}{\mathbf{K}} = \underset{p,p}{K_f} \otimes \underset{N,N}{K_x} \tag{32}$$

$$\underset{pN,p}{\mathbf{k}} = \underset{p,p}{K_f} \otimes \underset{N,1}{k(\mathbf{X}, \mathbf{x}_*)} \tag{33}$$

$$\underset{p,p}{\mathbf{k}_*} = \underset{p,p}{K_f} \otimes \underset{1,1}{k(\mathbf{x}_*, \mathbf{x}_*)}\,. \tag{34}$$

Prediction at a set of points

Multiple test points The emulator $\mathbf{f}$ is used to predict the output at a set of M new inputs $\mathbf{X}_*$, giving outputs $\mathbf{Y}_* = \mathbf{f}(\mathbf{X}_*)$, i.e. $(\mathbf{y}_{*1}, \ldots, \mathbf{y}_{*M}) = (\mathbf{f}(\mathbf{x}_{*1}), \ldots, \mathbf{f}(\mathbf{x}_{*M}))$. We denote $\tilde{\mathbf{y}}_*$ the vectorised set of test outputs. In the most general case, the predicted output admits a distribution $p(\tilde{\mathbf{y}}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{Y}) = \mathcal{N}(\mu_*, \mathbf{\Sigma}_*)$, with:

$$\underset{pM,1}{\mu_*} = \underset{pM,pN}{\mathbf{k}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,1}{\tilde{\mathbf{y}}} \tag{35}$$

$$\underset{pM,pM}{\mathbf{\Sigma}_*} = \underset{pM,pM}{\mathbf{k}_*} - \underset{pM,pN}{\mathbf{k}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,pM}{\mathbf{k}} \tag{36}$$

where $\mathbf{k}$ denotes the covariance between the test point and the training set, $\mathbf{K}$ denotes the training set's covariance and $\mathbf{k}_*$ is the covariance of the test point. We use the following notation:

$$\underset{pN,pM}{\mathbf{k}} = \underset{p,p}{K_f} \otimes \underset{N,M}{k(\mathbf{X}, \mathbf{X}_*)} \tag{37}$$

$$\underset{pM,pM}{\mathbf{k}_*} = \underset{p,p}{K_f} \otimes \underset{M,M}{k(\mathbf{X}_*, \mathbf{X}_*)}\,. \tag{38}$$

Separable case

For a separable emulator, the prediction equations can be rewritten using the following matrix identities (Petersen and Pedersen, 2008):

$$(A \otimes B)\,\mathrm{vec}(X) = \mathrm{vec}\left[B\,X\,A^{\mathrm{T}}\right] \tag{39}$$

$$A \otimes B + A \otimes C = A \otimes (B + C) \tag{40}$$

to give:

$$\mu_* = \mathrm{vec}\left[k(\mathbf{X}, \mathbf{X}_*)^{\mathrm{T}}\, K_x^{-1}\, \mathbf{Y}\right], \tag{41}$$

$$\mathbf{\Sigma}_* = K_f \otimes \left[k(\mathbf{X}_*, \mathbf{X}_*) - k(\mathbf{X}, \mathbf{X}_*)^{\mathrm{T}}\, K_x^{-1}\, k(\mathbf{X}, \mathbf{X}_*)\right]. \tag{42}$$

Note that μ_* is the vectorised form of the single output predicted mean from Equation (3) while the second term in the Kronecker product for $\mathbf{\Sigma}_*$ is the single output predicted covariance from Equation (4).

2.1.2 Estimation of the hyperparameters (separable case)

Marginal log-likelihood

The marginal log-likelihood is given by:

$$\ln p(\tilde{\mathbf{y}}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{-1}\tilde{\mathbf{y}} - \frac{1}{2}\ln|\mathbf{K}| - \frac{pN}{2}\ln 2\pi \tag{43}$$

$$\begin{aligned} &= -\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\mathrm{vec}\left[K_x^{-1}\mathbf{Y}K_f^{-1}\right] - \frac{N}{2}\log|K_f| \\ &\quad - \frac{p}{2}\log|K_x| - \frac{pN}{2}\ln 2\pi \end{aligned} \tag{44}$$

where we have used Equation (39) and the following identity (Petersen and Pedersen, 2008):

$$\log|K_f \otimes K_x| = \log\left(|K_f|^N |K_x|^p\right) \tag{45}$$
$$= N\log|K_f| + p\log|K_x| \tag{46}$$

Derivatives of MLL

If K_f and K_x have parameters θ_f and θ_x respectively and these parameters are disjoint, we can express the derivatives of the marginal log-likelihood as:

$$\frac{\partial}{\partial\theta_f}\ln p(\tilde{\mathbf{y}}|\mathbf{X},\boldsymbol{\theta}) = \frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\mathrm{vec}\left[K_x^{-1}\mathbf{Y}K_f^{-1}\frac{\partial K_f}{\partial\theta_f}K_f^{-1}\right] - \frac{N}{2}\mathrm{Tr}\left(K_f^{-1}\frac{\partial K_f}{\partial\theta_f}\right), \tag{47}$$

$$\frac{\partial}{\partial\theta_x}\ln p(\tilde{\mathbf{y}}|\mathbf{X},\boldsymbol{\theta}) = \frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\mathrm{vec}\left[K_x^{-1}\frac{\partial K_x}{\partial\theta_x}K_x^{-1}\mathbf{Y}K_f^{-1}\right] - \frac{p}{2}\mathrm{Tr}\left(K_x^{-1}\frac{\partial K_x}{\partial\theta_x}\right). \tag{48}$$

Note that, because of the separable nature of the covariance function, these derivatives only require inversion of the K_f and K_x matrices (not the full $\mathbf{K}$).

2.2 With mean function

Following Fricker (2008), the mean function can be represented by a set of univariate basis functions $\mathbf{h}$ with a matrix of regression coefficients B to be determined from the data.

If we assume that the same basis functions are shared by all outputs, the mean function for the set of points $\mathbf{h}(\mathbf{X})^{\mathrm{T}}B$ can be rewritten $\mathrm{H}^{\mathrm{T}}\tilde{\boldsymbol{\beta}}$ where $\tilde{\boldsymbol{\beta}} = \mathrm{vec}\,[B]$ and $\mathrm{H}^{\mathrm{T}} = \mathbf{I}_p \otimes H^{\mathrm{T}}$. $\mathbf{I}_p$ is the $p \times p$ identity matrix. We also use the notations $\mathrm{H} = \mathbf{I}_p \otimes H$ and $\mathrm{H}_*^{\mathrm{T}} = \mathbf{I}_p \otimes H_*^{\mathrm{T}}$.

2.2.1 Prediction (informative prior)

Prediction with mean function (prior)

If prior knowledge about the regression coefficients is available, i.e. $\tilde{\boldsymbol{\beta}} \sim \mathcal{N}(\mathbf{b},\mathbf{B})$, the equations for the predicted mean and variance are given by:

$$\underset{pM,1}{\mu'_*} = \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\ \underset{pq,1}{\bar{\beta}} + \underset{pM,pN}{\mathbf{k}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\left(\underset{pN,1}{\tilde{\mathbf{y}}} - \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\ \underset{pq,1}{\bar{\beta}}\right) \tag{49}$$
$$= \underset{pM,1}{\mu_*} + \underset{pM,pq}{\mathbf{R}^{\mathrm{T}}}\ \underset{pq,1}{\bar{\beta}} \tag{50}$$
$$\underset{pM,pM}{\boldsymbol{\Sigma}'_*} = \underset{pM,pM}{\boldsymbol{\Sigma}_*} + \underset{pM,pq}{\mathbf{R}^{\mathrm{T}}}\left(\underset{pq,pq}{\mathbf{B}^{-1}} + \underset{pq,pN}{\mathrm{H}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\right)^{-1}\ \underset{pq,pM}{\mathbf{R}} \tag{51}$$

where:

$$\underset{pq,pM}{\mathbf{R}} = \underset{pq,pM}{\mathrm{H}_*} - \underset{pq,pN}{\mathrm{H}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,pM}{\mathbf{k}}, \tag{52}$$
$$\underset{pq,1}{\bar{\beta}} = \left(\underset{pq,pq}{\mathbf{B}^{-1}} + \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pq,pN}{\mathrm{H}}\right)^{-1}\left(\underset{pq,pN}{\mathrm{H}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,1}{\tilde{\mathbf{y}}} + \underset{pq,pq}{\mathbf{B}^{-1}}\,\tilde{\boldsymbol{\beta}}\right). \tag{53}$$

Separable case

In the case of a separable emulator , we have the further identities:

$$\mathbf{R} = \mathbf{I}_p \otimes \left[H_* - HK_x^{-1}\mathbf{k}\right] = \mathbf{I}_p \otimes R, \tag{54}$$
$$\mathrm{H}^{\mathrm{T}}\mathbf{K}^{-1}\mathrm{H} = K_f^{-1} \otimes \left[HK_x^{-1}H^{\mathrm{T}}\right] \tag{55}$$
$$\mathrm{H}\mathbf{K}^{-1}\tilde{\mathbf{y}} = \mathrm{vec}\left[HK_x^{-1}\mathbf{Y}K_f^{-1}\right] \tag{56}$$

However, this is not sufficient to obtain a fully separable expression of $\bar{\beta}$. Even if one chose a separable prior over $\tilde{\boldsymbol{\beta}}$, so that $\mathbf{B} = \mathrm{B}^f \otimes \mathrm{B}_x$, one would still need to inverse a $pq \times pq$ matrix in the computation of $\bar{\beta}$, thus limiting the benefits of using a separable emulator. Only if the separable prior was chosen such that $\mathbf{B} = K_f \otimes \mathrm{B}_x$ would a separable expression of $\bar{\beta}$ be obtained. However, there is no reason why one should want to constraint $\mathbf{B}$ on K_f, especially since K_f is likely to be estimated from the data (while the prior covariance matrix $\mathbf{B}$ should not).

Using a prior on $\bar{\beta}$ breaks the separable assumption

2.2.2 Prediction (uninformative prior)

For an uninformative prior on $\tilde{\boldsymbol{\beta}}$, i.e. $\mathbf{B}^{-1} \to 0$, the predicted distribution is obtained by extending the single output case to multivariate outputs. This gives the predicted mean and variance:

Prediction with mean function (no prior)

$$\underset{pM,1}{\mu'_*} = \underset{pM,pq}{\mathrm{H}_*^{\mathrm{T}}}\ \underset{pq,1}{\bar{\beta}} + \underset{pM,pN}{\mathbf{k}^{\mathrm{T}}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ (\ \underset{pN,1}{\tilde{\mathbf{y}}} - \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\ \underset{pq,1}{\bar{\beta}}\ ) \tag{57}$$

$$= \underset{pM,1}{\mu_*} + \underset{pM,pq}{\mathbf{R}^{\mathrm{T}}}\ \underset{pq,1}{\bar{\beta}} \tag{58}$$

$$\underset{pM,pM}{\mathbf{\Sigma}'_*} = \underset{pM,pM}{\mathbf{\Sigma}_*} + \underset{pM,pq}{\mathbf{R}^{\mathrm{T}}}\ (\ \underset{pq,pN}{\mathrm{H}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\ )^{-1}\ \underset{pq,pM}{\mathbf{R}} \tag{59}$$

where $\bar{\beta}$ is now given by

$$\underset{pq,1}{\bar{\beta}} = (\ \underset{pq,pN}{\mathrm{H}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,pq}{\mathrm{H}^{\mathrm{T}}}\ )^{-1}\ \underset{pq,pN}{\mathrm{H}}\ \underset{pN,pN}{\mathbf{K}^{-1}}\ \underset{pN,1}{\tilde{\mathbf{y}}}\ . \tag{60}$$

The expression for $\bar{\beta}$ factorises in the case of a separable emulator, as follows:

Separable case

$$\bar{\beta} = \mathrm{vec}\left[\left(HK_x^{-1}H^{\mathrm{T}}\right)^{-1} HK_x^{-1}\mathbf{Y}\right] = \mathrm{vec}\left[\bar{B}\right]. \tag{61}$$

This leads to the fully separable prediction equations:

$$\mu'_* = \mathrm{vec}\left[H_*^{\mathrm{T}}B + k(\mathbf{X}, \mathbf{X}_*)^{\mathrm{T}}K_x^{-1}(\mathbf{Y} - HB)\right] \tag{62}$$

$$= \mu_* + \mathrm{vec}\left[R^{\mathrm{T}}B\right] \tag{63}$$

$$\mathbf{\Sigma}'_* = \mathbf{\Sigma}_* + K_f \otimes \left[R^{\mathrm{T}}(HK_x^{-1}H^{\mathrm{T}})^{-1}R\right]. \tag{64}$$

2.2.3 Estimation of the parameters (informative prior)

The marginal log-likelihood is given by:

MLL with mean function (prior)

$$\begin{aligned}\ln p(\tilde{\mathbf{y}}|\mathbf{X}, \boldsymbol{\theta}, \mathbf{b}, \mathrm{B}) = &-\frac{1}{2}(\mathrm{H}^{\mathrm{T}}\mathbf{b} - \tilde{\mathbf{y}})^{\mathrm{T}}(\mathbf{K} + \mathrm{H}^{\mathrm{T}}\mathbf{B}\mathrm{H})^{-1}(\mathrm{H}^{\mathrm{T}}\mathbf{b} - \tilde{\mathbf{y}}) \\ &-\frac{1}{2}\ln|\mathbf{K} + \mathrm{H}^{\mathrm{T}}\mathbf{B}\mathrm{H}| - \frac{pN}{2}\ln 2\pi \end{aligned} \tag{65}$$

$$\propto -\frac{1}{2}\,\mathrm{d}^{\mathrm{T}}G^{-1}\mathrm{d} - \frac{1}{2}\ln|G| \tag{66}$$

where

$$\mathrm{d} = \mathrm{H}^{\mathrm{T}}\mathbf{b} - \tilde{\mathbf{y}}, \tag{67}$$

$$G = \mathbf{K} + \mathrm{H}^{\mathrm{T}}\mathbf{B}\mathrm{H}. \tag{68}$$

2.2.4 Estimation of the parameters (uninformative prior)

In the case of an uninformative prior, the marginal log-likelihood is given by:

$$\begin{aligned}\ln p(\tilde{\mathbf{y}}|\mathbf{X},\boldsymbol{\theta}) = &-\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\mathbf{K}^{-1}\tilde{\mathbf{y}} - \frac{1}{2}\log|\mathbf{K}| \\ &+\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\mathbf{C}\tilde{\mathbf{y}} - \frac{1}{2}\log|\mathbf{A}| - \frac{p(n-m)}{2}\log 2\pi,\end{aligned} \tag{69}$$

where:

$$m = \text{rank}(H), \tag{70}$$

$$\mathbf{A} = \mathrm{H}\mathbf{K}^{-1}\mathrm{H}^{\mathrm{T}} \tag{71}$$

$$\mathbf{C} = \mathbf{K}^{-1}\mathrm{H}^{\mathrm{T}}\mathbf{A}^{-1}\mathrm{H}\mathbf{K}^{-1}. \tag{72}$$

In the separable case, we can rewrite $\mathbf{A}$ and $\mathbf{C}$:

$$\mathbf{A} = K_f^{-1} \otimes \left[HK_x^{-1}H^{\mathrm{T}}\right] \qquad = K_f^{-1} \otimes A, \tag{73}$$

$$\mathbf{C} = K_f^{-1} \otimes K_x^{-1}H^{\mathrm{T}}A^{-1}HK_x^{-1} \qquad = K_f^{-1} \otimes C, \tag{74}$$

and the marginal log-likelihood becomes:

$$\begin{aligned}\ln p(\tilde{\mathbf{y}}|\mathbf{X},\boldsymbol{\theta}) \propto &-\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\text{vec}\left[(K_x^{-1}-C)\mathbf{Y}K_f^{-1}\right] - \frac{N-q}{2}\log|K_f| \\ &-\frac{p}{2}\log|K_x| - \frac{p}{2}\log|A|\end{aligned} \tag{75}$$

The derivatives follow easily:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}_f}\ln p(\tilde{\mathbf{y}}|\mathbf{X},\boldsymbol{\theta}) = &\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\text{vec}\left[(K_x^{-1}-C)\mathbf{Y}K_f^{-1}\frac{\partial K_f}{\partial\theta_f}K_f^{-1}\right] \\ &-\frac{N-q}{2}\mathrm{Tr}\left(K_f^{-1}\frac{\partial K_f}{\partial\theta_f}\right),\end{aligned} \tag{76}$$

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}_x}\ln p(\tilde{\mathbf{y}}|\mathbf{X},\boldsymbol{\theta}) = &\frac{1}{2}\tilde{\mathbf{y}}^{\mathrm{T}}\text{vec}\left[\left(K_x^{-1}\frac{\partial K_x}{\partial\theta_x}K_x^{-1} + \frac{\partial C}{\partial\theta_x}\right)\mathbf{Y}K_f^{-1}\right] \\ &-\frac{p}{2}\mathrm{Tr}\left(K_x^{-1}\frac{\partial K_x}{\partial\theta_x} + A\frac{\partial A}{\partial\theta_x}\right).\end{aligned} \tag{77}$$

The derivatives of C and A are obtained from Equations (28) and (29).

2.3 Conclusion

We have presented a summary of emulation, giving the key results for emulator with and without mean function, with univariate and multivariate output. In particular, the separable assumption on the covariance function for the multivariate case provides an interesting framework in which the computations can be kept manageable by involving matrix operations in input space or output space only, rather than in the much larger joint input/output space. We note that several desirable features such as the use of a prior on the regression coefficients in the mean function or the addition of a noise (nugget) term in the covariance function break the separable assumption. This can be seen as a limitation of the separable emulator framework.

In the following section, we look at the application of separable emulators to the case where the simulator is a dynamic model.

3 Emulation of dynamic models

3.1 Overview

Dynamic models are a specific class of computer models which simulate the evolution in time of the *state* of a system. Such a simulator takes as an input the state of the system at a time t and outputs the state at a later time $\mathrm{t} + \mathrm{dt}$: $\mathbf{x}_{\mathrm{t+dt}} = \boldsymbol{\eta}(\mathbf{x}_{\mathrm{t}}, \mathrm{dt})$

Several approaches to emulating a dynamic simulator can be found in Conti and O'Hagan (2007):

- Emulate the trajectory of the state given its initial condition. The input is the state at a time t and the output is the time-series $\{\mathbf{x}_{\mathrm{t+dt}}, \mathbf{x}_{\mathrm{t+2dt}}, ..., \mathbf{x}_{\mathrm{t}+\tau\mathrm{dt}}\}$. This method provides an efficient way to compute the trajectory of the state (single prediction step) but can be computationally demanding depending on the dimension of the output ($p\tau$).
- Emulate the transition from $\mathbf{x}_{\mathrm{t}}$ to $\mathbf{x}_{\mathrm{t+dt}}$. This keeps the dimension of the problem relatively low. A time-series of future states can still be generated by repeated application of the emulator from some initial condition.
- Emulate time and state jointly. Although the most flexible in theory, this approach is extremely inefficient in practice due to the impractical dimension of the joint state time space.

In this study, we focus on building an emulator for a single time step of the simulator (method 2), for a fixed dt, i.e. the input space is the space spanned by $\mathbf{x}_{\mathrm{t}}$ and the output space is the space spanned by $\mathbf{x}_{\mathrm{t+dt}}$.

3.2 Emulation of the Lorenz '63 model

The Lorenz '63 model is a 3-variable non-linear chaotic model. The evolution of the state is governed by the set of differential equations:

$$\frac{\mathrm{d}x_1}{\mathrm{dt}} = \sigma(x_2 - x_1) \tag{78}$$

$$\frac{\mathrm{d}x_2}{\mathrm{dt}} = x_1(\rho - x_3) - x_2 \tag{79}$$

$$\frac{\mathrm{d}x_3}{\mathrm{dt}} = x_1 x_2 - \beta x_3 \tag{80}$$

where x_i denotes the ith dimension of $\mathbf{x}$. The parameters σ, ρ and β are set to their default values, respectively 10, 28 and 8/3.

3.2.1 Design considerations

Like many dynamic models, the L63 model Lorenz (1963) presents attractors, meaning that repeated application of the simulator generates a state trajectory which does not explore the whole state space, but only a subspace of it we refer to as the manifold (i.e. the subset of possible "realistic" values taken by $\mathbf{x}_{\mathrm{t}}$ as a result of propagating it forward in time). We thus want to restrict our training set

to points lying on the manifold. To estimate that manifold, we consider a long run of the model from some initial condition and discard the initial part of the trajectory during which the state has not yet converged to the manifold ("burn in" phase).

Initial experiments have looked at training sets sampled uniformly (in time) from the state's trajectory. However, better schemes exist if we believe that the changes undergone by the state during a single time step vary in different regions of the state space. For instance, the L63 model has two attractors, resulting in state either orbiting around one of the attractors or transiting from one attractor to the other. These transition phases are typically harder to capture with an emulator ; it thus makes sense to use a training set in which more observations (i.e. training points) are made in the regions where transitions occur and fewer in the more stable regions where the state orbits around one of the attractors.

Unfortunately, we are not able to identify such regions prior to running the simulator. An alternative method consists in adding points to the training set in regions where the variance, as predicted by the emulator, is maximal, i.e. where we know the least about the simulator. The parameters of the emulator are re-estimated by maximising the marginal likelihood of the data every time a new design point is added (for a small number of iterations only, e.g. 10).

In this experiment, the model is integrated forward with an inner time step of 0.01 time units, for a total of 800 time steps (8 time units). We look at emulating a fixed time step $\mathrm{dt} = 0.05$ time units with a multivariate separable emulator. The emulator has a linear mean function and a squared exponential covariance function with parameters to be determined from the data. Our training set is generated by selecting 30 design points as explained previously. The resulting training set is shown on Figure 1 (dots) along with the underlying simulator run (dashed line).

Figure 3 shows, on the left plot, how the predicted covariance decreases as the number of points increases (in log-space). The covariance is averaged over time and dimensions. At the end of the training phase, the covariance is about 10^{-4}. On the right hand side, the estimation of the length scale parameters (in log space) is plotted against the number of training points. We observe that there is a minimum training size (around 15 in this example) below which the emulator acts in a very "conservative" way (small length scales), while above that threshold, it becomes more confident in the data and starts interpolating smoothly (large length scales). Although this behaviour can be expected, it is worth noting that the transition between the conservative and smooth regimes is not even, but rather discontinuous.

Can we get a theoretical estimate of this threshold?

3.2.2 One step ahead prediction using the emulator

Figure 2 shows the same information as Figure 1, along with the emulator's prediction (mean: solid black line, covariance: dashed black line). Figure 4 shows the same information but the output is plotted against time. It is almost impossible to distinguish between the emulator and the simulator, and the predicted covariance remains very small over the whole run. Note that 200 time steps (from 6 time units onwards) have been kept aside and left unobserved in order to assess the quality of emulator outside the training set. The emulator shows very good prediction skill in that area, which suggests it has learnt the model. However, al-

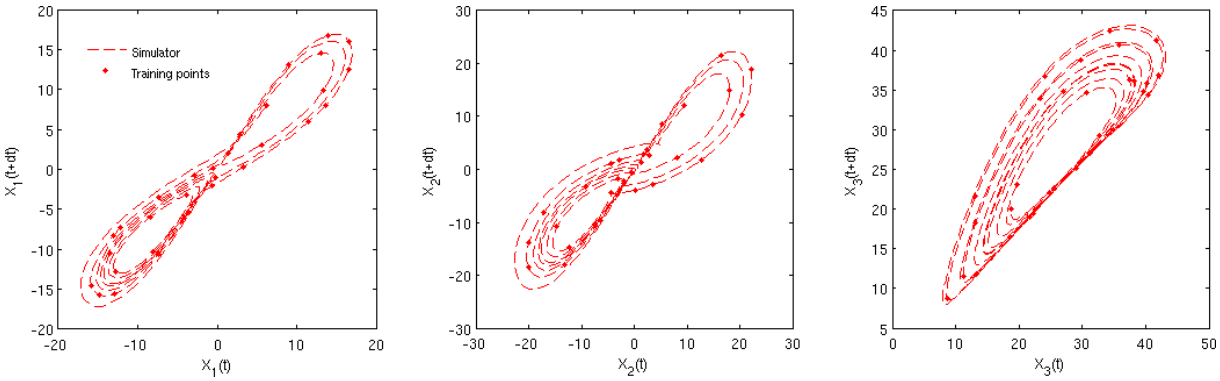

Figure 1: Lorenz '63 – A single run of the model generating the well-known "butterfly" trajectory (dashed line). Training points (dots) are selected using a minimum predicted variance approach.

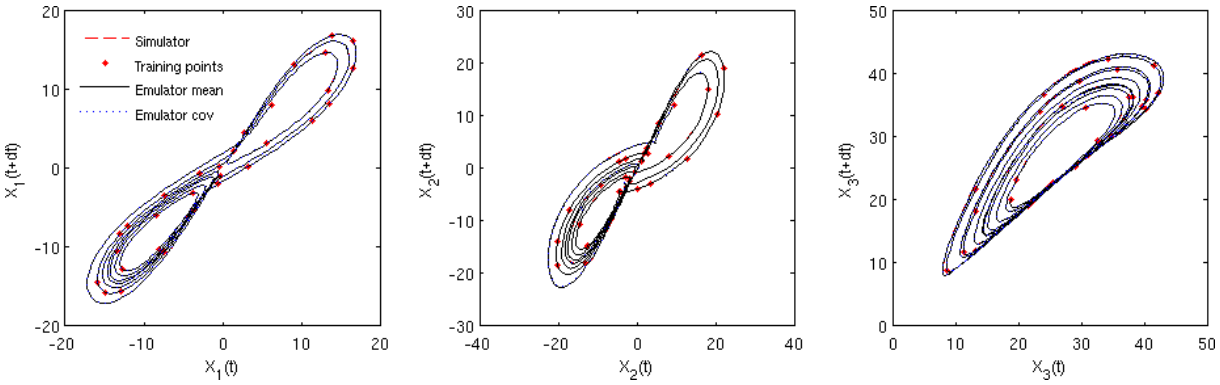

Figure 2: Lorenz '63 – The same data as in Figure 1, plotted along with the emulator's mean response (black line) and predicted variance (dotted line). The emulator's prediction is indistinguishable from the true output.

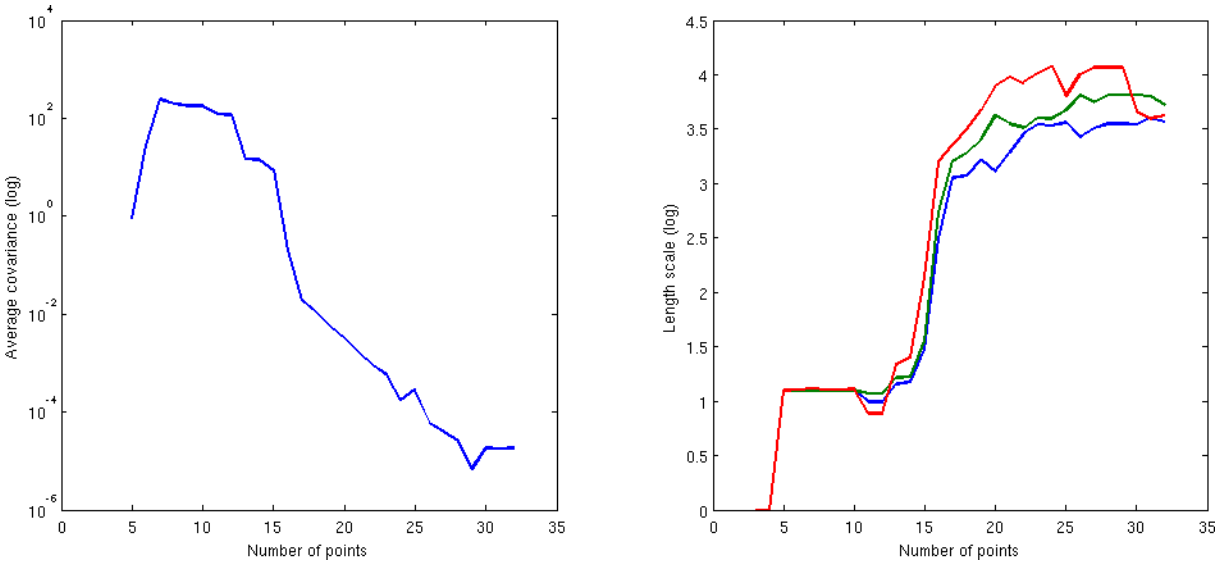

Figure 3: Lorenz '63 – Average predicted variance (left) and length-scale parameters (right) as the number of design points increases. The predicted variance is averaged over dimension and time. Both the variance and the length-scales are plotted in log scale.

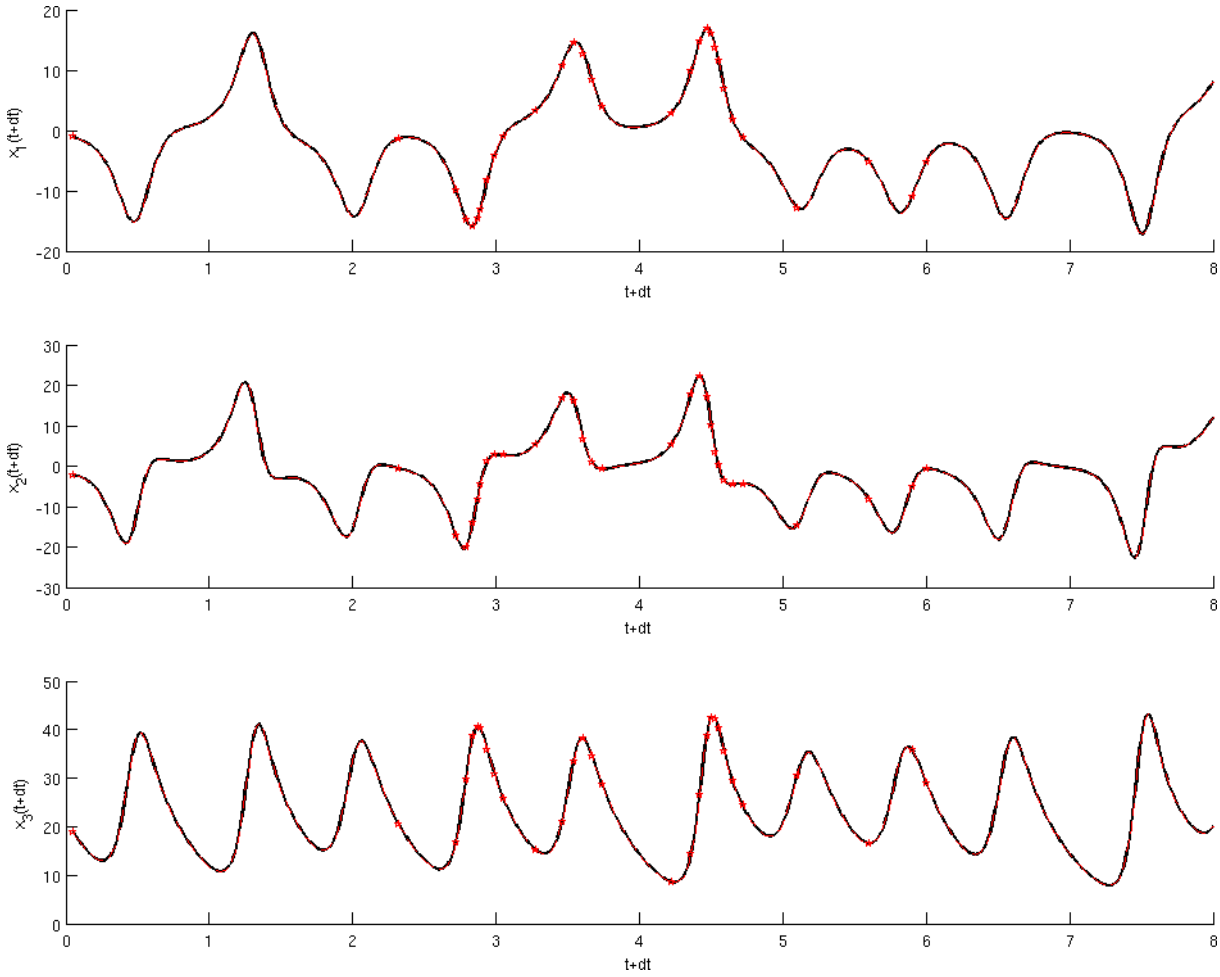

Figure 4: Lorenz '63 – Emulated trajectory (projected onto each dimension). Given a true trajectory of the state at time t, the emulator is used to compute the future trajectory at $\mathrm{t} + \mathrm{dt}$ (i.e. each point on the trajectory is propagated forward using the emulator). The simulated trajectory at time $\mathrm{t} + \mathrm{dt}$ is shown as a dashed red line, while the emulated mean trajectory is shown as a solid black line. Note that the black line shadows the red line almost perfectly, with variance almost zero everywhere. Red stars denote the observed points used to train the emulator.

though this area is unobserved in time, it is likely to be close to observed data in input space, hence conclusions about the emulation skill when extrapolating (in time) must be taken with caution.

Is there a better way to select an unobserved region away from the training data in space rather than in time? This is what we really want in order to assess the emulator's extrapolation skill.

3.2.3 Sequential prediction using the emulator

Propagation of the state forward in time is achieved by sequential application of the forward model to the state. In order to assess the emulator's ability to propagate the state forward, we can similarly apply the emulator repeatedly to the state. However, the uncertainty introduced by the emulator needs to be propagated too. This can be done in two ways: using approximate analytic methods, whereby an estimate of the propagated uncertainty is derived, or using Monte-Carlo methods, whereby a sample of realisations is propagated through the emulator and the propagated uncertainty is estimated from the sample. We resort to the latter method as it is conceptually simpler, though more demanding computationally. Propagation of the uncertainty is summarised in the following algorithm:

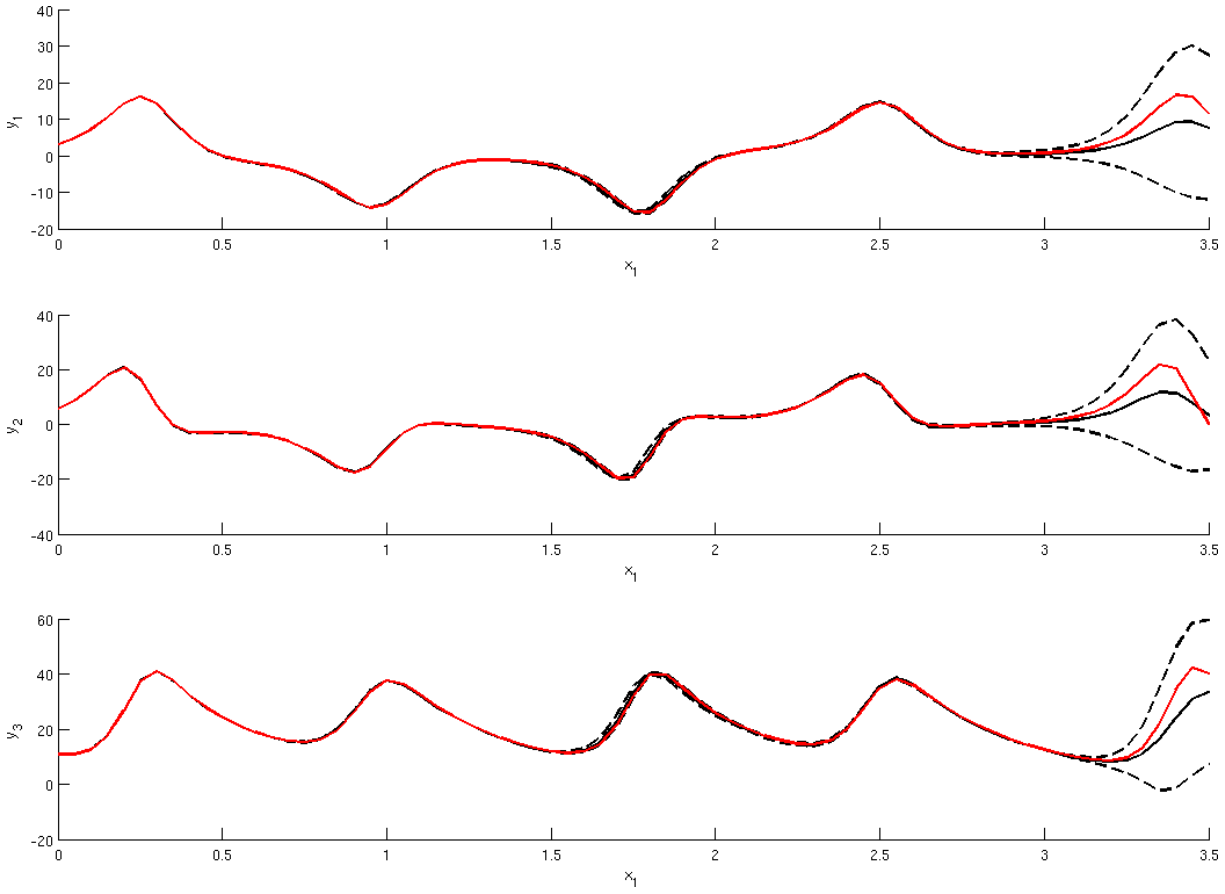

Figure 5: Lorenz '63 – Sequential emulation

1. Select an initial condition $\mathbf{x}_{\mathrm{t}} = \mathbf{x}_0$
2. Propagate $\mathbf{x}_{\mathrm{t}}$ through the emulator, giving the predicted distribution $\mathrm{p}(\mathbf{x}_{\mathrm{t+dt}})$
3. Sample N realisations $\mathbf{x}^i_{\mathrm{t+dt}}$ from $\mathrm{p}(\mathbf{x}_{\mathrm{t+dt}})$
4. At each time t, propagate each sample $\mathbf{x}^i_{\mathrm{t}}$ through the emulator, giving a predicted distribution $\mathrm{p}(\mathbf{x}^i_{\mathrm{t+dt}})$
5. Sample $\mathbf{x}^i_{\mathrm{t+dt}}$ from $\mathrm{p}(\mathbf{x}^i_{\mathrm{t+dt}})$
6. Repeat steps 4 and 5 to desired lead time

It is clear that the performance of the emulator for sequential prediction is strongly related to the initial condition $\mathbf{x}_0$ chosen, and to the chosen sample of realisations if its size is small. In order to take these considerations into account when assessing the emulator's performance, we need to average over several initial conditions, and possibly several samples. The latter can be avoided by choosing a sample of large enough size. In the following experiment, we propagate the state forward in time from 50 different initial conditions (taken from the same true simulator trajectory the data was generated from), and chose a sample size of 100 which we believe is sufficient for sampling effects to be negligible. Figure 5 shows the mean trajectory of the state from a given initial condition ($\mathrm{t}_0 = 1$ time unit) along with 2 standard deviation error bars, as estimated from the sample.

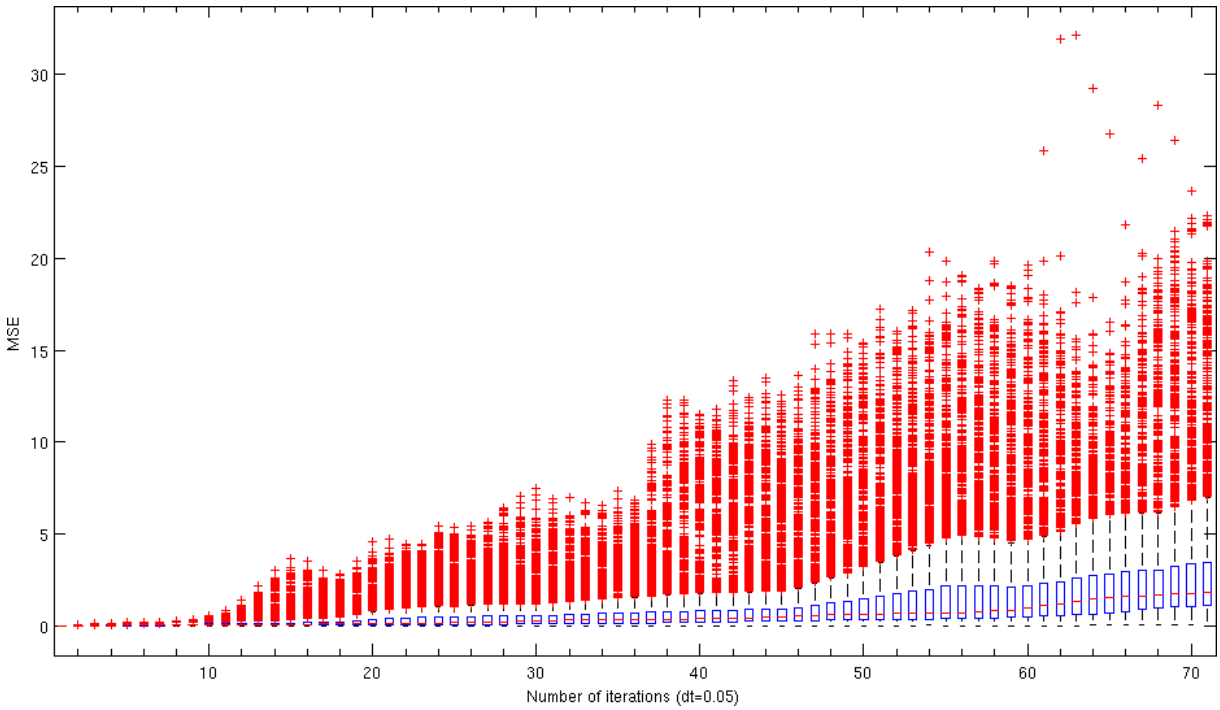

Figure 6: Lorenz '63 – Sequential emulation of $\mathbf{x}_{t+dt} = \mathbf{f}(\mathbf{x}_t)$. This plot shows the distribution of the Mean Square Error between the simulated and emulated trajectories as a function of increasing lead time. 100 sample trajectories are computed using the emulator, from 50 different initial conditions (5000 trajectories in total). The distribution of the error is calculated over these 5000 trajectories.

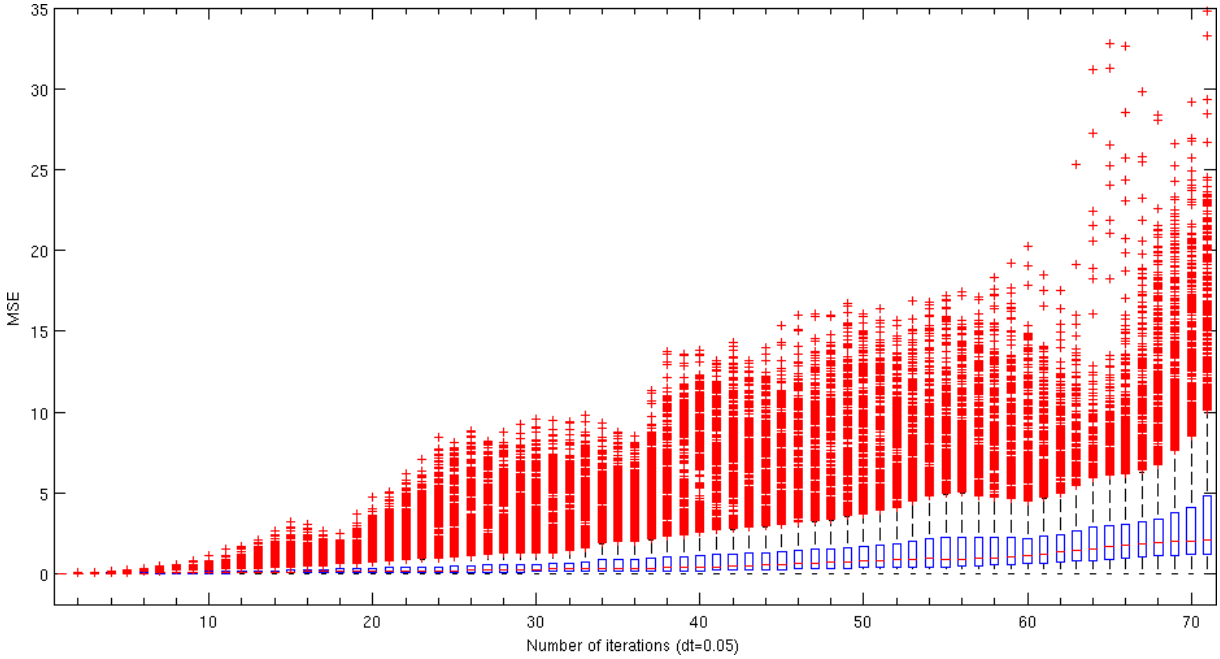

Figure 7: Sequential emulation of $\mathbf{x}_{t+dt} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t)$. This plot shows the same information as Figure 6 for the case where the variation in $\mathbf{x}_t$ is emulated rather than $\mathbf{x}_{t+dt}$ itself.

We then measure, for each sample trajectory, the Mean Square Error (MSE) to the true trajectory (as given by the simulator). Figure 6 summarises the distribution of the MSE as a function of lead time (from 1 to 71 time steps in the ahead, with 20 steps corresponding to 1 time unit). As expected, the error increases as a function of lead time but remains, for most samples, within acceptable bounds up to 50 time steps (2.5 time units) ahead. This demonstrates that the emulator is able to reproduce the true dynamics of the simulator, although having seen only 30 simulator runs. There is, however, an important number of outliers, showing that the emulator fails to produce realistic trajectories for some choices of initial conditions and samples. This is a limitation due to the approximations performed in the emulator, but which could be improved on by increasing the number of training points.

The same experiment was repeated, but this time the change in $\mathbf{x}_\mathrm{t}$, i.e. $\Delta\mathbf{x}_\mathrm{t} = \mathbf{x}_{\mathrm{t}+\mathrm{dt}} - \mathbf{x}_\mathrm{t}$ is considered, rather than the propagated state $\mathbf{x}_{\mathrm{t}+\mathrm{dt}}$. Figure 7 shows that emulating $\Delta\mathbf{x}_\mathrm{t}$ leads to similar results to emulating $\mathbf{x}_{\mathrm{t}+\mathrm{dt}}$.

3.2.4 Further design considerations

The emulator has been shown to give good results when trying to emulate a single run of the model. If one knows the state of the system at an initial time, then because the system is deterministic, considering a single run of the model from that initial condition is a sensible approach. However, there might be parts of the system space that this unique trajectory does not explore, and the emulator can be expected to perform poorly when run from a different realistic, but unobserved, initial condition. We want to compare the performace of two emulators, one trained on a single model run, and one trained on several model runs from different initial conditions, in order to compare their performances when generalising to unobserved initial conditions. We choose two emulators, each trained using a design set of 100 points. The first emulator selects its design points from a single run of 8000 time steps of the model. The second selects its design points from 10 runs of the model, each 800 time steps long (so that both emulators are given the same amount of data to select from). We then compare their performance from 50 different initial conditions (selected at random from another 10 runs of the model). Figure 8 shows the distribution of the Mean Square Error for the first (top) and second (bottom) designs. In this case, the second design causes the emulator to fail to reproduce the model's dynamics. The first design shows good, consistent results. In particular, the MSE remains below 40, which corresponds roughly to the true model's amplitude and suggests that the trajectories generated, although wrong, are still consistent with the model's dynamics.

3.3 Emulation of the Lorenz '96 model

Our next aim is dynamic emulation of the Lorenz 96 (L96) model Lorenz (1996). The L96 model is a 40-variable model representing 40 atmospheric sensors located around a latitude. The system is governed by the set of equations:

$$\frac{\partial \mathbf{x}_i}{\partial \mathrm{t}} = (\mathbf{x}_{i+1} - \mathbf{x}_{i-2})\mathbf{x}_{i-1} - \mathbf{x}_i + F \tag{81}$$

indicating that the observation at each sensor is linked to the observations at the 3 closest neighbours. F is a constant forcing term.

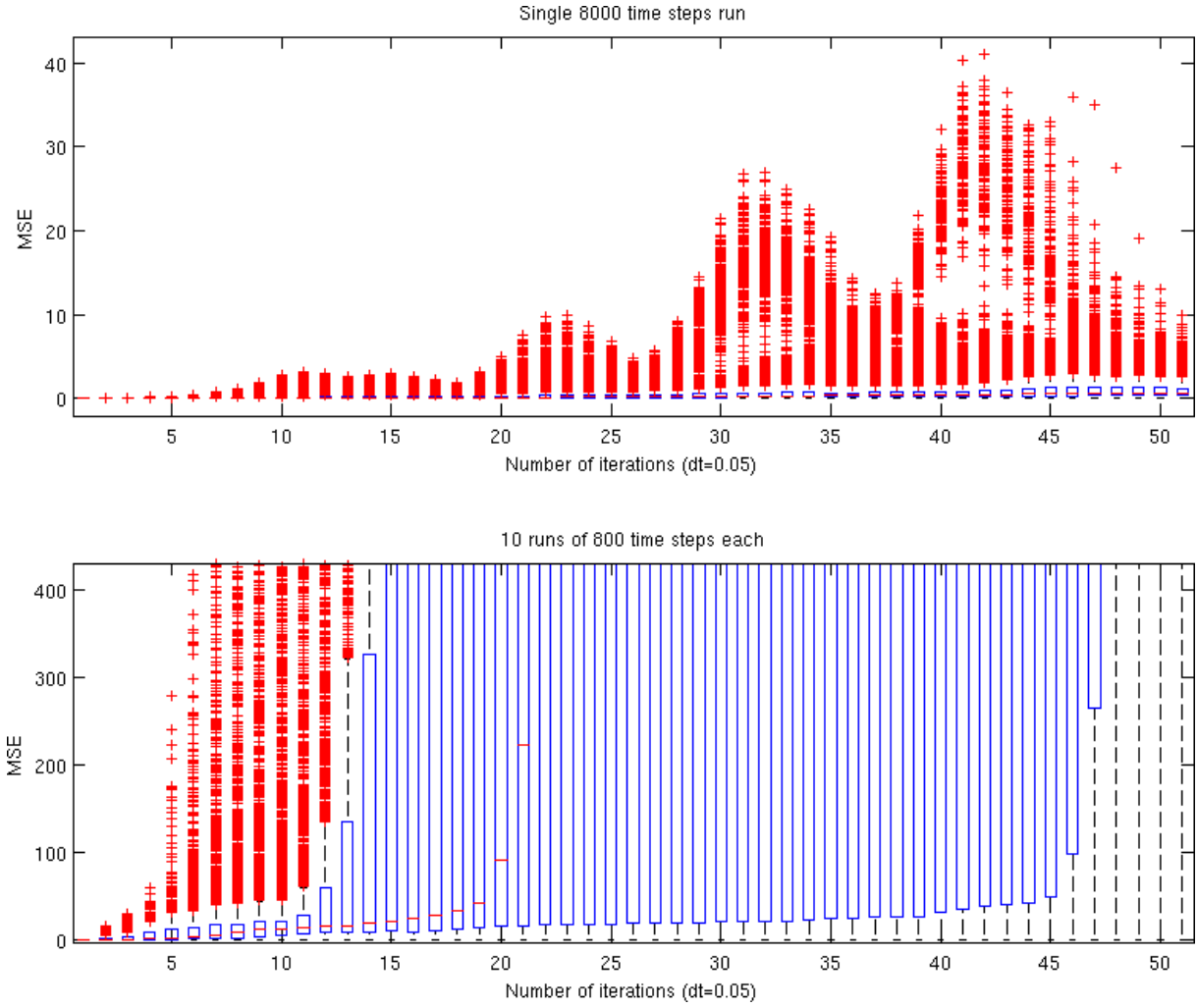

Figure 8: Lorenz '63 – Comparison of sequential emulation for 2 different designs. The top plot shows the Mean Square Error to the simulator for a design based on a single long run of the simulator. The bottom plot shows the same information for a design based on several short runs of the simulator from slightly different conditions.

Our initial aim was the application of dimension reduction methods, in particular Principal Components Analysis (PCA), to the emulation of the L96 model. Figure 9 shows the variance of $\Delta \mathbf{x}_t$ as provided by PCA (normalised). There is no clear cut in the eigenspectrum, which suggests that very few components can be neglected when emulating the system. Similar observations can be made when applying PCA to the input data.

However, it is questionable whether considering the input or the output separately is sensible in the first place. What we are really interested in is the dimension of the mapping from input to output. In the worst case, this dimension is $p \times p$, i.e. every dimension of the input space affects every dimension of the output space. Yet, we know that only a subset of the dimension of $\mathbf{x}$ is responsible for the output $\Delta \mathbf{x}^i$. Further work will look at Canonical Correlation Analysis, where both the input and the output are considered.

Preliminary work on the L96 model focused on emulating a single run of the model. A sequence of input points $\mathbf{x}_t$ is generated by running the model forward from some initial condition for 1000 time steps with an internal time step of 0.01

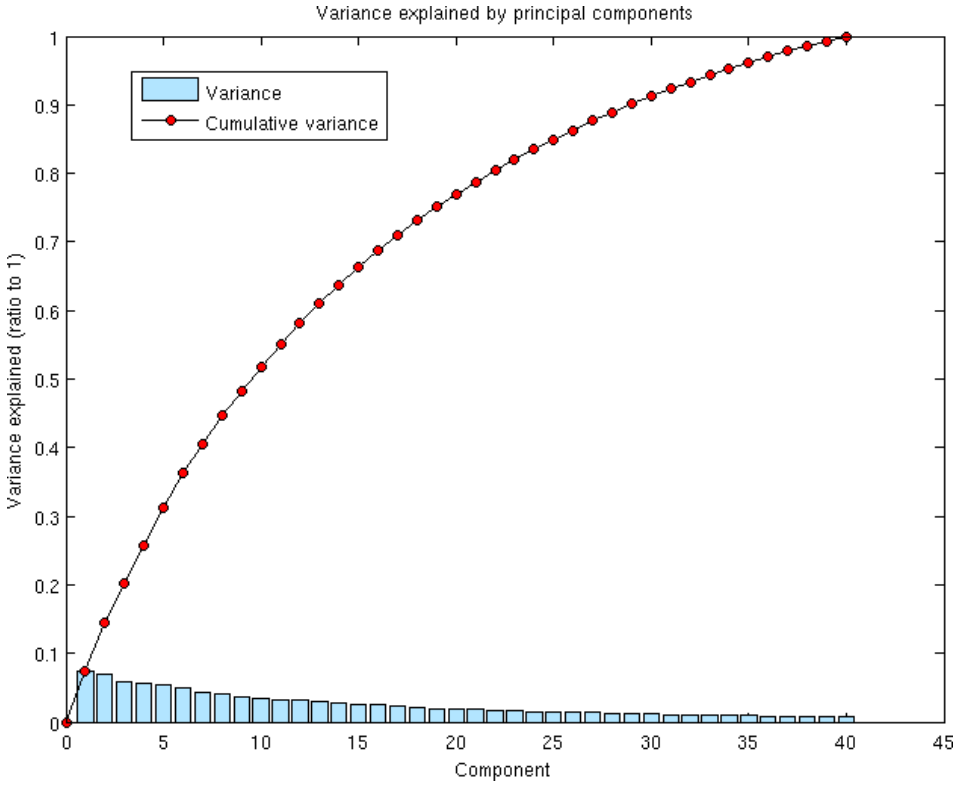

Figure 9: Lorenz '96 – Principal components analysis of the output data

time units. For each of these points, we consider the output $\Delta \mathbf{x}_t = \mathbf{x}_{t+dt} - \mathbf{x}_t$ where $dt = 0.05$ time units. 80 design points are selected from the first 800 inputs using a minimum predicted variance approach, keeping the last 200 inputs (i.e. the last 2 time units) unobserved to assess the emulator's extrapolation skill. The output covariance K_f is considered diagonal for simplicity, although future experiments will look at improved covariance structures. The covariance on the input uses an isotropic squared exponential kernel. Using a single length scale parameter for all dimensions is justified by the symmetric nature of the L96 model.

Figure 10, left, shows the average covariance per input as the size of the design set increases. Figure 10, right, shows the length scale parameter (solid blue line) and the first two output variances (dashed lines) as a function of the number of design points. Here again, note that we observe a critical threshold of about 75-80 points necessary for the predicted variance to drop and for the covariance parameters to converge. Note that in this experiment, the parameters were re-estimated every 5 new design points only (for speed purposes).

Figure 11 shows the output of the emulator plotted against time, with projections onto the 5 first dimensions shown from top to bottom. The emulator (solid black line) closely matches the simulator (red dashed line) in the observed region, but quickly diverges from it in the unobserved region. Further experiments need to look at larger training sets from longer runs of the model, as it is clear that on this short example, the design chosen does not explore the 40 dimensional space properly.

These preliminary results are to be extended with more experiments in order to understand better the issues of dynamic emulation in high dimensions and the

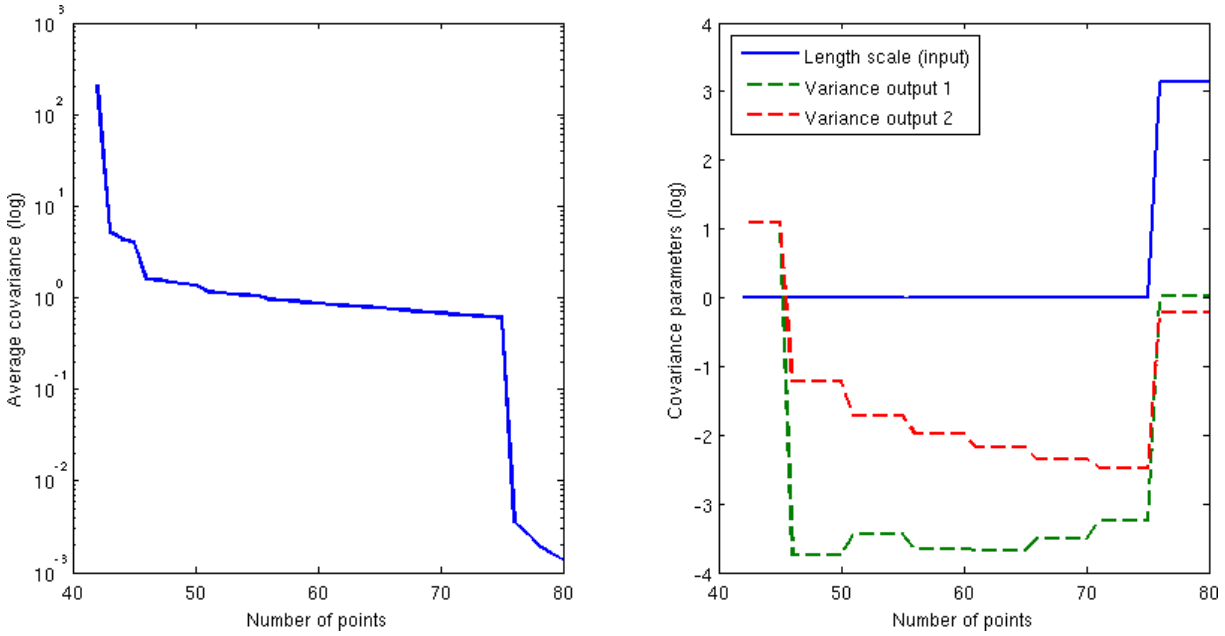

Figure 10: Lorenz '96 – Average predicted variance (left) and covariance parameters (right). Both are plotted in log scale as a function of the number of design points.

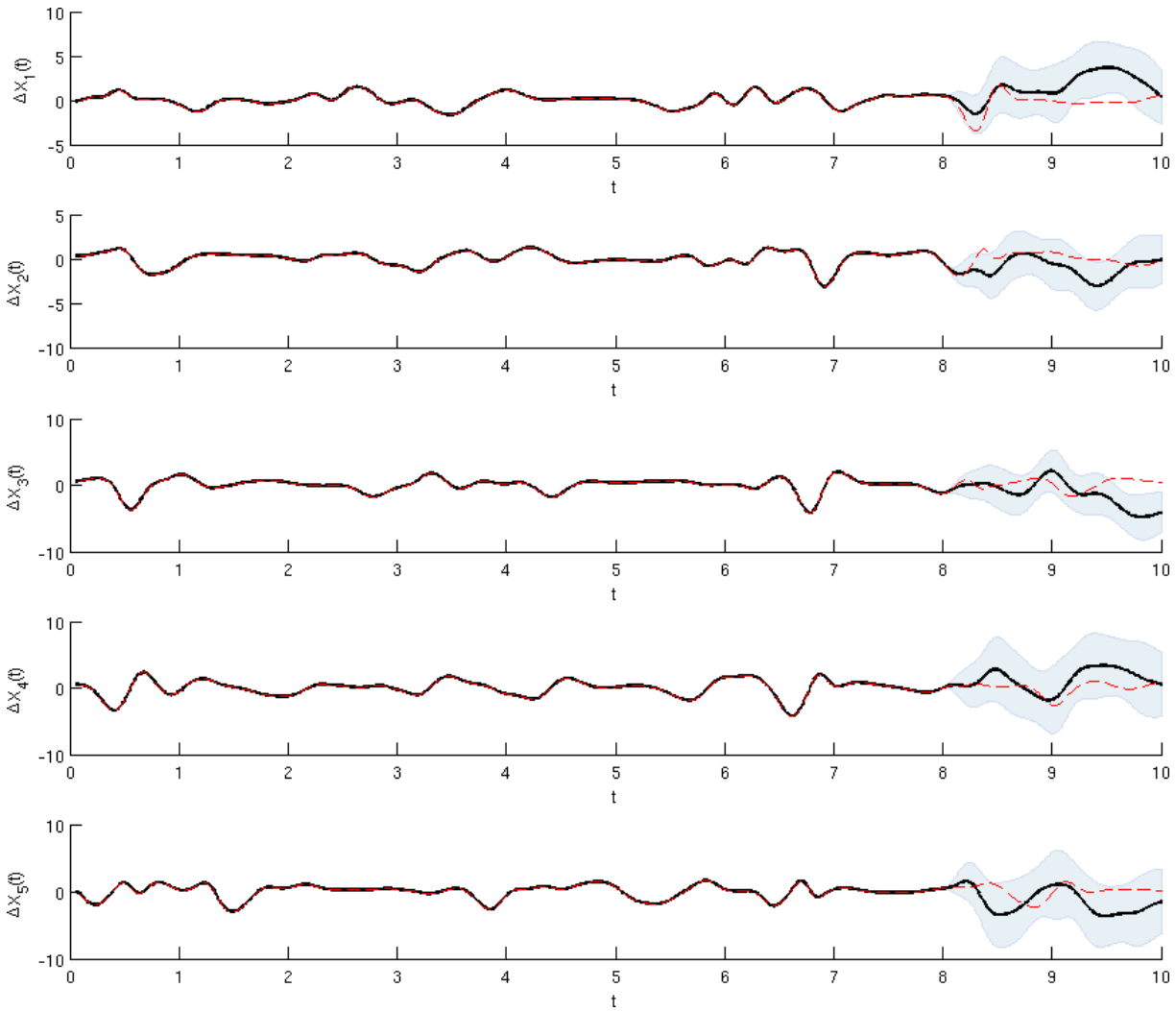

Figure 11: Lorenz '96 – Emulation of a single model trajectory

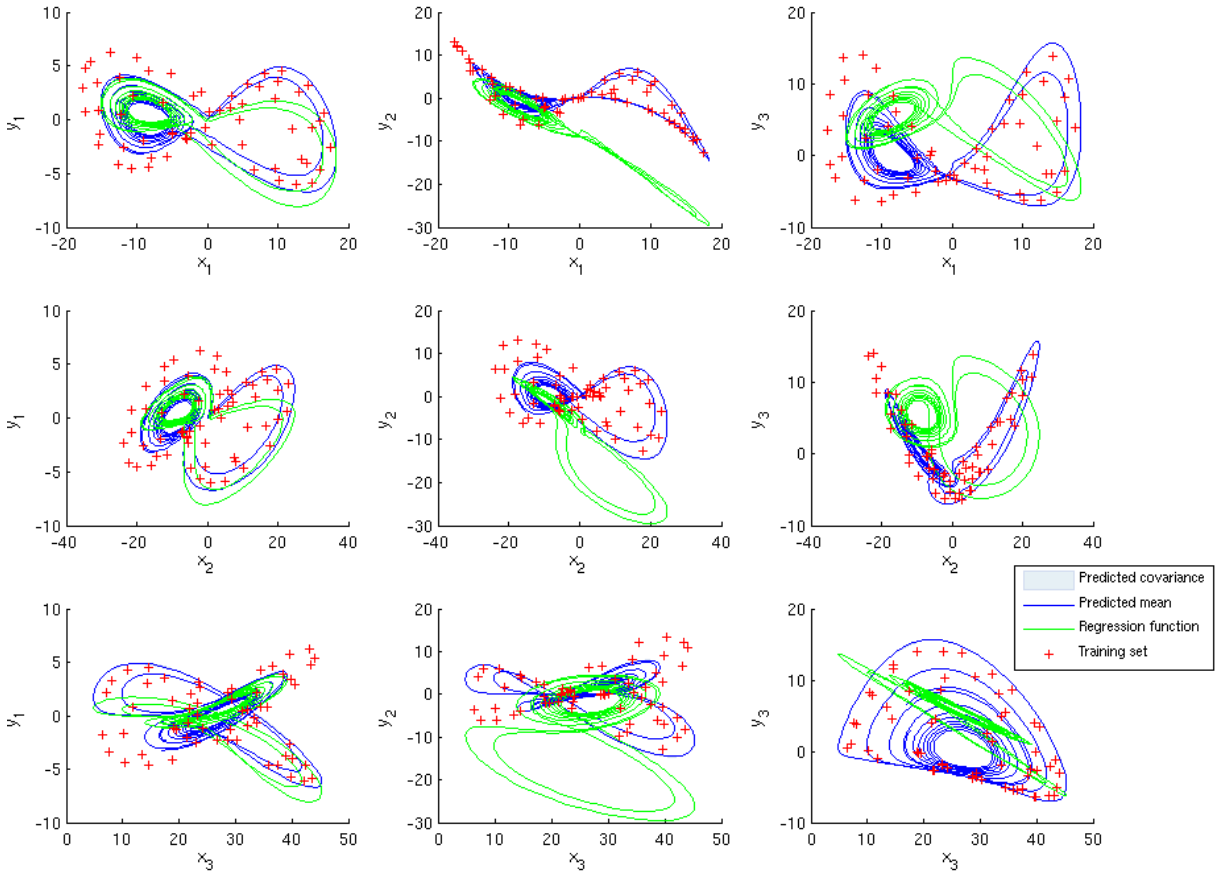

Figure 12: Mean function issue: the mean function (green) seems to be "pushed away" from the data (red crosses). The prediction (in blue) is made at points taken from a run of the model (hence the trajectory line). Each plot shows the output ($\Delta \mathbf{x}_t$) against the input ($\mathbf{x}_t$), projected onto each of the 3 input dimensions (left to right) and each of the 3 output dimensions (top to bottom).

potential benefits of dimension reduction methods in that context.

3.4 Open questions

3.4.1 Issue with mean function

The following phenomenon has been observed when training the emulator: when the size of the design set becomes sufficiently large and the emulator's response becomes very close to the simulator's, the mean function seems to be "pushed away" from the data, as seen on Figure 12. Although this is of little concern in well observed regions of input space, where the Gaussian process accounts for the discrepancy, it is important in unobserved regions, where the emulator defaults to the mean function.

This issue seems to be related to the fact that the uncertainty in the emulator's response becomes small, and accordingly the length scale parameters in the covariance function become large. However, it is unclear at this stage exactly why this causes the regression coefficients H to be badly estimated. One possible reason is that both large length scales and large data sets could induce large cross-correlations between data points, leading to an ill-conditioned K_x matrix. Further investigation to better understand the mean function problem is critical (these

could include, for example, monitoring the condition number of the K_x matrix as the emulator's confidence increases).

A possible solution to prevent the mean function from being shifted away from the data would be to systematically centre the data (i.e. substract the mean) and get rid of the constant terms in the linear mean function. Another option is to constrain the coefficients of the constant term to remain small via the use of an appropriate prior. A third, possibly better option, would consist in using a prior on the length scale parameters to prevent them from getting to large (i.e. use maximum a posteriori instead of the maximum likelihood when estimating these parameters). This could also help achieving a smoother estimation of these parameters, rather than the jump observed with maximum likelihood (Figure 3, right plot).

3.4.2 Unbounded uncertainty with mean function

A second remark, linked to the previous, has to do with predicted variance. It is easy to see, from the expression of the variance in Equation (64) that as $\mathbf{x}_*$ moves away from the data, its predicted variance is dominated by the second term, which is proportional to the square of H_* through R (all the rest being fixed with respect to $\mathbf{x}_*$). In consequence, if the mean function uses unbounded regressors such as polynomials, the predicted variance is unbounded too. This also applies to the predicted mean, which can be problematic when emulating a physical system evolving within a bounded domain (as many physical system do). Note that this phenomenon can be aggravated by poor estimation of the regression coefficients (as a result of the previous remark).

To address this issue, one might want to put a prior on the regression coefficients β to ensure they remain small. However, eliciting such a prior in high-dimension can be difficult. Furthermore, using an informative prior on the β partially breaks the separability of the emulator and could raise computational costs considerably.

3.4.3 Multiple minima for marginal likelihood

A last remark concerns the estimation of the covariance parameters. We have observed, in many cases, that when a large number of parameters need to be estimated, the marginal likelihood presents several local minima, and the "optimal" value of the parameters can depend strongly on their initial value. This is more noticeable in high dimension, where we try to estimate the full K_f matrix ($p(p+1)/2$ parameters). Although this is the most flexible approach, it is also the most expensive, and a good choice of initial parameters is critical. This is less the case when using simpler covariance structures, such as band diagonal and diagonal (the latter being equivalent to emulating along each dimension of the output independently). However, prior specification of the output covariance structure might only be possible in cases where the simulator is known (i.e. computer model).

4 Conclusions

This report aimed at providing a clear account of the past and current work on dimension reduction methods applied to emulation of dynamic models. It is clear that, at this stage, most of the work has focused on learning about the emulator

framework (Sections 1 and 2) and applying it to simple dynamic models (Section 3). This has provided some insight into several unexpected issues (Section 3.4), which are to be explored further in future developments. The dimension reduction aspect has only been briefly outlined, but it is the focus of ongoing work and more detailed results will be provided in a future report.

The author is particularly grateful to Alexis Boukouvalas for his proof-reading of, and useful comments on, this report.

References

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

E.V. Bonilla, K.M.A. Chai, and C.K.I. Williams. Multi-task Gaussian Process Prediction. In *Advances in Neural Information Processing Systems*. MIT Press, 2007.

S. Conti and A. O'Hagan. Bayesian emulation of complex multioutput and dynamic computer models. *Research report*, 569(07), 2007.

T. Fricker. Multiple-Output Emulators: First Year PhD report. Technical report, January 2008.

Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Science*, 20:130–141, 1963.

Edward N. Lorenz. Predictability – A problem partly solved. *Proc. seminar on predictability*, 1996.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, oct 2008. URL `http://matrixcookbook.com`. Version 20081110.

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

A Notations

This section summarises the notation used in this work. The symbols and their explanations are given along with their LaTeX command name (available on the repository as a style file `gp.sty`).

A.1 General notations

Symbol	LaTeX	Meaning	Dimension
n	`\dimin`	Dimension of input space	
p	`\dimout`	Dimension of output space	
$\boldsymbol{\eta}$	`\GPsimul`	Simulator/Model	$n \to p$
$\mathbf{f}$	`\GPemul`	Emulator with zero mean	$n \to p$
$\mathbf{g}$	`\GPemulmf`	Emulator with mean function	$n \to q$

A.2 Training set

Symbol	LaTeX	Meaning	Dimension
N	`\dimtrn`	Size of training set	
$\mathbf{x}$	`\GPin`	Training input	n,1
$\mathbf{y}$	`\GPout`	Training output	p,1
$\mathbf{X}$	`\GPIn`	Training design matrix	n,N
$\mathbf{Y}$	`\GPOut`	Training set output	p,N
$\tilde{\mathbf{y}}$	`\GPOutVec`	Vectorised training set output	pN
k	`\covfin`	Covariance function on the input	
K_x	`\covmin`	Covariance between input training points	N,N
K_f	`\covmout`	Covariance between output components	p,p

A.3 Test set

Symbol	LaTeX	Meaning	Dimension
M	`\dimtst`	Size of test set	
$\mathbf{x}_*$	`\GPinTest`	Test input	n,1
$\mathbf{y}_*$	`\GPoutTest`	Test output	p,1
$\mathbf{X}_*$	`\GPInTest`	Test set input	n,M
$\mathbf{Y}_*$	`\GPOutTest`	Test set output	p,M
$\tilde{\mathbf{y}}_*$	`\GPOutTestVec`	Vectorised test set output	pM
$\mathbf{K}$	`\GPCovmTrain`	Covariance of training set	pN,pN

A.4 Prediction

Symbol	LaTeX	Meaning	Dimension
μ_*	`\GPmeanPred`	Mean of single predicted output	p,1
$\mathbf{\Sigma}_*$	`\GPcovmPred`	Covariance of single predicted output	p,p
$\mathbf{k}$	`\GPcovmTrainTest`	Covariance between training set and test point	pN,p
$\mathbf{k}_*$	`\GPcovmTest`	Covariance of test point	p,p
μ_*	`\GPMeanPred`	Mean of predicted output set	pM
$\mathbf{\Sigma}_*$	`\GPCovmPred`	Covariance of predicted output set	pM,pM
$\mathbf{k}$	`\GPCovmTrainTest`	Covariance between training and test sets	pN,pM
$\mathbf{k}_*$	`\GPCovmTest`	Covariance of test set	pM,pM

A.5 Mean function

Symbol	LaTeX	Meaning	Dimension
$\mathbf{h}(\mathbf{x})$	`\gpmf(\GPin)`	Mean function basis functions	q,1
β	`\gpmfcoeff`	Mean function regression coefficients	q,1
$\mathbf{b}$	`\gpmfcoeffmean`	Prior on regression coefficients - mean	q,1
B	`\gpmfcoeffcovm`	Prior on regression coefficients - covariance	q,q
$\bar{\beta}$	`\gpmfcoeffpred`	Optimal regression coefficients - mean	q,1