Fault detection based on Gaussian process latent variable models

J. Serradilla[a], J.Q. Shi[a,*], A.J. Morris[b]

[a]*School of Mathematics and Statistics, Newcastle University, Newcastle upon Tyne, NE1 7RU, UK*
[b]*School of Chemical Engineering and Advanced Materials, Newcastle University, Newcastle upon Tyne, NE1 7RU, UK*

Abstract

Gaussian processes, $\mathcal{GP}$s, are routinely used to approximate complex non-linear functions with relative simplicity. Their regression performance is, at least, comparable to that achieved via artificial neural networks (ANN) and, in fact, both methods are intrinsically related. They are both non-parametric and, as Neal (1994) has shown, when the number of nodes in the hidden layer of a neural network tends to infinity the ANN converge to a Gaussian process.

In most of the cases, the $\mathcal{GP}$ will map a multivariate input into a univariate response. In this paper, however, we present an approach to process monitoring that combines *independent and identically distributed* $\mathcal{GP}$s so that multivariate responses can be appropriately modeled. We review a similar approach that has already been proposed in the literature and highlight some concerns related to it that must taken into consideration. Additionally, we offer an alternative approach to the way that new observations are mapped into the non-linear model. A simple simulated example is provided that will help understand the method flexibility. Furthermore, results from a real example are also discussed.

Keywords: Latent Variable Models, Process Monitoring, Multivariate Statistical Process Control, MSPC, Gaussian process

1. Notation

The reader familiar with MSPC will notice that our use of notation throughout the paper is different to the common standard found in the field. We denote the data set of N observations on D variables as $\mathbf{Y}$ instead of $\mathbf{X}$. We also refer to the variables in the Q-dimensional latent space as $\mathbf{X}$ instead of $\mathbf{T}$. Although this is a matter of personal preference, the change should bring about more clarity in keeping with the standard terminology used in regression problems.

2. Introduction

There are two main approaches to process modeling. On the one hand, models can be built based on the underlying physics and chemistry laws that govern the behavior of the process; this is referred to as *mechanistic modeling* and requires a thorough and extensive knowledge about the system under study. Very often, restrictions both in term of cost and time will simply prevent their development. On the other hand, a viable alternative is to use the data that is routinely collected from the process to build a *data-based* model. Whereas these models are much easier to develop, it is also true that the information that can extracted from them is rather more limited. In many instances, the data-based methodology is used as a *black-box* where the user expects to extract a reliable prediction of how the system is behaving without having to worry about the inner-working of the true generative process.

When it comes to *data-based* fault detection and diagnosis, one could further subclassify the models into three main groups:

- *No-model*, where the individual process variables are monitored directly.
- *Linear models*, where the underlying assumption is that the process is linear and some form of linear surrogate variable is formed and subsequently monitored.

*Corresponding author. Email address: j.q.shi@ncl.ac.uk.

- *Non-linear models*, where the assumption is that the process is non-linear and non-linear auxiliary variables are constructed and then monitored.

It is undoubtedly very appealing to simply not build a model and monitor the process variables individually. This is an ideal situation as fault detection is almost instantaneous and fault diagnosis is direct in the sense that the variable moving outside its confident limits is the variable developing a fault. But this situation is not practical: today's manufacturing processes measure and log hundreds of variables and therefore individual variable monitoring is unrealistic to say the least; it besides ignores the fact that the correct functioning of the process depends on the *joint behaviour* of a set of variables and not on each variable individually (Kourti and MacGregor, 1995). Attempts can be made to remove the inessential variables and choose a subset of the original variables that contain, according to a specific criterion, as much information as possible. This form of variable selection was first introduced by McCabe (1984). Exploiting this idea Srinivasan and Qian (2007) have shown how a multi-state process could be monitored by just focusing on those variables whose behaviour is essential for the smooth running of the process; these variables most important from a monitoring perspective were termed as *key variables* by the authors. While the key-variable approach tackles the issue of dimensionality reduction via variable selection, it does not consider the problem of variable association that could lead to potential departures from normal plant behaviour.

Using linear models to build combinations of the original variables in order to reduce the problem dimensionality and, at the same time, obtain a valid representation of the process is also an attractive idea. These multivariate projection methods, as they are commonly known, are built around principal component analysis, PCA, and the numerous variants thereof. For simplicity, in this paper we will refer to these linear methods simply as PCA. The concept behind PCA is to project the original data, which includes noise and redundant variables, into a latent space with the objective of capturing the true dimensionality of the system (Wold et al., 1987). The method is not based on a probability model although it has a probabilistic interpretation which stems from a linear factor analysis model with isotropic Gaussian noise (Tipping and Bishop, 1999). PCA is very efficient at building a fingerprint of normal plant behaviour which can, by comparison, be used at a later stage for fault detection and diagnosis. Most chemical engineering systems, however, are non-linear and therefore models accounting for that non-linearity would be the most appropriate. Of course, they are many examples of successful applications of PCA to non-linear systems and arguments in favour of doing that; see, for instance, Kourti (2002). Nevertheless, in most cases, when doing so, the model will simply be used as a black-box or dimensionality-reduction artifact where the number of principal components retained has no resemblance with the real underlying dimensionality of the problem. A very clear example of this is given by Simoglou et al. (2000), who managed to identify a problem in an industrial system by looking at principal components that were explaining very little of the total variance in the system covariance matrix.

In this paper, we use a non-linear approach to extracting the underlying characteristics of the process. The backbone of the procedure is the *Gaussian process latent variable*, GPLV, model first suggested by Lawrence (2005) within the machine learning community. The idea is simply to consider *independent and identically distributed* $\mathcal{GP}$s to map the input space variables, $\boldsymbol{x} \in \mathcal{R}^Q$, into the observational space, $\boldsymbol{y} \in \mathcal{R}^D$. Note that *a-priori* the input positions $\boldsymbol{x}$ are unknown and therefore need to be determined. In a second step, when new observations become available, we make use of two neural networks to project them first onto the latent space and subsequently onto the original observational space. This approach shares similarities to the non-linear principal component analysis based on principal curves, NLPCA, developed by Dong and McAvoy (1996). Let $\mathbf{Y} \in \mathcal{R}^{N\times D}$ be our original observations and $\mathbf{X} \in \mathcal{R}^{N\times Q}$ the corresponding latent variable representation. Dong and McAvoy's approach relies on an additive model, i.e.

$$\mathbf{Y} = \sum_{i=1}^{Q} f_i(x_i) + \text{Error}, \quad f_i \text{ is non linear}$$

which assumes that the original observations are generated as a linear combination of $Q-$univariate non-linear functions; the latent variables can therefore be determined one at a time. The GPLV model, on the other hand, is not restricted to additive models and can account for multiplicative effects as all the latent variables are determined simultaneously. The GPLV

model is also closely related to the concept of Input-Training neural network, IT-net, proposed by Tan and Mavrovouniotis (1995). The idea is that the net input variables are not fixed but adjusted along with internal network parameters so that it can reproduce the net output more efficiently. Jia et al. (1998) show how a process fault can successfully be detected using the IT-met to map the latent variables into the observations that have been compressed via PCA. A significant advantage of using the GPLV model over the IT-net is that it requires a substantially lower number of parameters; it is also a full probabilistic model where prediction uncertainty and hypothesis testing can be carried out if necessary.

Ge and Song (2010) have recently shown how the GPLV model can be used in process monitoring. However, their approach to dealing with new observations can be problematic if it is not performed carefully. We aim to explain in this paper where our concerns lie when it comes to projecting new observations onto the GPLV model and offer an alternative that deals with the problem. Prior to defining the GPLVM in Section 4 we first offer a quick introduction to $\mathcal{GP}$s in Section 3. We then describe Ge and Song's approach to projecting new observations onto the model as well as our alternative in Section 5. Finally, both a simple simulation example and a real application are given in Section 7.

3. Gaussian processes

A short summary about $\mathcal{GP}$s is provided in this section. The interested reader should refer to Rasmussen and Williams (2006, Chapter 2), where the topic is discussed in detail.

3.1. $\mathcal{GP}$ priors

Let us consider the data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i),\ \boldsymbol{x}_i \in \mathcal{R}^Q, y_i \in \mathcal{R}\}_{i=1}^N$, i.e. it comprises N pairs of observations each consisting of a *Q-dimensional* input vector $\boldsymbol{x}_i$ and a scalar output y_i. Let also $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^\intercal$ be the $N \times Q$ design matrix with all the input vectors and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\intercal$ the corresponding output vector. The $\mathcal{GP}$ regression model is defined as follows

$$\begin{aligned} y_i &= f(\boldsymbol{x}_i) + \varepsilon_i \\ \varepsilon_i &\sim \mathcal{N}(0, \sigma^2) \quad iid \\ f(\cdot) &\sim \mathcal{GP}(0, k(\cdot,\cdot)) \end{aligned} \tag{1}$$

In other words, we are assuming that y_i is related to $\boldsymbol{x}_i$ non-linearly through an unknown function f, which, in turn, it is being approximated by a $\mathcal{GP}$. And by saying that the function f follows a $\mathcal{GP}$ it is meant that, over the finite range of input observations $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, the vector $\boldsymbol{f} = (f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n))^\intercal$ follows a multivariate normal distribution. This distribution is commonly specified as having mean zero and a $N \times N$ covariance matrix generated via $k(\cdot,\cdot)$, the *covariance function* or *kernel*.

The kernel allows to write the covariance between the noise-free output, $f(\boldsymbol{x}_i)$, as a function of the input vectors, $\boldsymbol{x}_i$. It is a key part of the $\mathcal{GP}$ as it will govern the properties of the regressed function and it must always generate a positive semi-definite covariance matrix. In this paper we use the *squared exponential* kernel defined as

$$\begin{aligned} k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \operatorname{cov}\left(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)\right) \\ &= \nu_0 \exp\left\{ -\frac{1}{2} \sum_{q=1}^{Q} \gamma (x_{iq} - x_{jq})^2 \right\} \end{aligned} \tag{2}$$

where (ν_0, γ) are unknown parameters . Let us now define $\mathbf{K}$ as the *covariance* or *kernel matrix* evaluated at all pairs of the N training observations, i.e. $\mathbf{K} = (k_{ij})$.

3.2. $\mathcal{GP}$ posterior

It can be shown that the marginal distribution of the output vector $\boldsymbol{y}$ follows a multivariate normal distribution

$$\boldsymbol{y} \sim \mathcal{N}_N(\mathbf{0}, \mathbf{K}_y = \mathbf{K} + \sigma^2 \mathbf{I}) \tag{3}$$

where $\mathbf{K}_y$ is the $N \times N$ covariance matrix whose $(i,j)^{th}$ element is defined as

$$(\mathbf{K}_y)_{ij} = \operatorname{cov}(y_i, y_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sigma^2 \delta_{ij} \tag{4}$$

with δ_{ij} being the Kronecker delta. Notice the subtle but important difference between $\mathbf{K}$, the noise-free covariance matrix, and $\mathbf{K}_y$ which incorporates the functional noise.

Finally, let us also define $\boldsymbol{\theta} = (\nu_0, \gamma, \sigma^2)$ as the vector of all unknown parameters. $\boldsymbol{\theta}$ contains both, the covariance function parameters,(ν_0, γ), and the functional noise, σ^2; it is commonly referred to as *hyper-parameter* vector to emphasize that the parameters are from a non-parametric model.

3.3. $\mathcal{GP}$ prediction

$\mathcal{GP}$s also provide a straightforward framework to predict the output $f(\boldsymbol{x}^*)$ for a new input vector $\boldsymbol{x}^*$. The joint distribution of the new enlarged vector of outputs $(y_1, y_2, \ldots, y_N, f(\boldsymbol{x}^*))^\intercal$ will still be multivariate normal; the prediction, i.e. $\hat{y}^*$, of $f(\boldsymbol{x}^*)|\mathcal{D}$ is a normal distribution whose mean and variance are given as

$$\begin{aligned} E(f(\boldsymbol{x}^*)|\mathcal{D}) &= \boldsymbol{k}^{*\intercal}\mathbf{K}_y^{-1}\boldsymbol{y} \\ \text{var}\,(f(\boldsymbol{x}^*)|\mathcal{D}) &= k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}^{*\intercal}\mathbf{K}_y^{-1}\boldsymbol{k}^* \end{aligned} \tag{5}$$

where $\boldsymbol{k}^* = (k(\boldsymbol{x}^*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}^*, \boldsymbol{x}_n))^\intercal$ is the vector of covariances between the new input point, $\boldsymbol{x}^*$, and the training data $\boldsymbol{x}_i$, $i = 1, \ldots, N$.

With the distribution of the training data known as given by Eq. (3), the log-likelihood function can be easily written as

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}_y| - \frac{1}{2}\boldsymbol{y}^\intercal\left(\mathbf{K}_y\right)^{-1}\boldsymbol{y}$$

Training of a Gaussian process involves determining the values of the unknown hyper-parameter vector $\boldsymbol{\theta}$ given the observed data, $\mathcal{D}$. That can be carried out by maximizing the previous log-likelihood function in a procedure known as *Empirical Bayes estimation*. Alternatively, a full Bayesian approach is also possible whereby prior distributions are allocated to each of the unknown parameters and are subsequently combined with the likelihood function. Full implementation details are provided by Shi and Choi (2011, Chapter 3).

4. Gaussian process latent variable models

Up to this point we have assumed that $\boldsymbol{x}_i$, $i = 1, \ldots, N$ were known and the aim of the inference process was to determine $\boldsymbol{\theta}$. It could be the case, however, that the input vectors $\boldsymbol{x}_i$ are unknown (i.e. latent); then, the purpose of the inference procedure would not only be to determine the best value of $\boldsymbol{\theta}$ but also the best value of the latent input positions that would maximize the likelihood of the observed data.

Consider a new dataset $\mathcal{D}$ comprising N D-dimensional observations, i.e. $\mathcal{D} = \{\boldsymbol{y}_i|_{i=1}^N,\ \boldsymbol{y}_i \in \mathcal{R}^D\}$. Instead of a collection of N-observations, the dataset can also be thought of a collection of D-variables, i.e. $\mathcal{D} = \{\boldsymbol{y}_{(d)}|_{d=1}^D,\ \boldsymbol{y}_{(d)} \in \mathcal{R}^N\}$. A Gaussian process latent variable model (Lawrence, 2005) is defined as

$$\boldsymbol{y}_{(d)}|\mathbf{X}, \boldsymbol{\theta} \overset{\text{ind}}{\sim} \mathcal{GP}(\mathbf{0}, k(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})) \tag{6}$$

where by $\overset{\text{ind}}{\sim}$ we mean *independent* and *identically distributed (iid)*. Therefore, this class of models are simply mappings, using *iid* $\mathcal{GP}$s, between $\mathbf{X}$, the $N \times Q$ latent space, and each output dimension $\boldsymbol{y}_{(d)}$.

4.1. GPLVM inference

Training of the GPLV model is the procedure whereby both the latent variables, $\mathbf{X}$, and the kernel hyper-parameters, $\boldsymbol{\theta}$, are determined. In order to do that, firstly, the joint joint marginal distribution for $\mathbf{Y}$, the $N \times D$ matrix of observations, can be written as

$$p(\mathbf{Y}|\mathbf{X}, \theta) \sim \prod_{d=1}^{D}\mathcal{N}(\mathbf{0}, \mathbf{K}_y)$$

where $p()$ denotes the probability density function. The associated log-likelihood can then be expressed as

$$\ell(\mathbf{X}, \boldsymbol{\theta}; \mathbf{Y}) = -\frac{D}{2}\log|\mathbf{K}_y| - \frac{1}{2}\,\text{tr}(\mathbf{K}_y^{-1}\mathbf{Y}\mathbf{Y}^\intercal) \tag{7}$$

where the constant terms have been omitted. Maximization of the previous function is, however, not possible without additional identifiability constraints. By giving a Gaussian prior distribution to each latent variable, $\boldsymbol{x}_i \sim \mathcal{N}(0, \mathbf{I}_Q)$, then $\mathbf{X} \sim \prod_{i=1}^N \mathcal{N}(0, \mathbf{I}_Q)$. Hence

$$p(\mathbf{X}) \propto \exp\left\{-\frac{1}{2}\text{tr}(\mathbf{X}\mathbf{X}^\intercal)\right\}$$

and the posterior distribution is given by:

$$p(\mathbf{X}, \theta|\mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \theta)p(\mathbf{X}) \tag{8}$$

We can then calculate the *maximum a posteriori* (MAP) solution with respect to the latent factor scores, $\mathbf{X}$, and the unknown parameters, $\boldsymbol{\theta}$, by maximizing the following log-likelihood

$$\ell(\mathbf{X}, \boldsymbol{\theta}; \mathbf{Y})_{MAP} = \ell(\mathbf{X}, \boldsymbol{\theta}; \mathbf{Y}) - \frac{1}{2}\,\text{tr}(\mathbf{X}\mathbf{X}^\intercal) \tag{9}$$

where constant terms have been omitted.

4.2. Empirical Bayes solution

A solution of the GPLV model can be found by jointly maximizing Eq. (9) with respect to $\mathbf{X}$ and $\boldsymbol{\theta}$. The model log-likelihood is both non-linear and *non-convex*. Due to the high-dimensionality of the problem, a global solution cannot be guaranteed and multiple local maxima will occur. As it is common in these cases, we randomly start the algorithm at different points and select the solution with the highest likelihood.

Between the array of non-linear optimizers that we can be used, *conjugate gradient methods* (Nocedal and Wright, 2006, Section 5.2) have been the suggested choice in the numerical analysis community when dealing with these specific problems. In broad terms, the *conjugate gradient method with line search (CGL)* works by iteratively computing search directions which are conjugate with respect the Hessian matrix (or an approximation thereof). Once the search direction has been found, a unidimensional line search with respect to the step size is carried out along the conjugate direction in order to determine a new approximation to the local minimum of the objective function. Note that conjugate gradient method avoid having to provide to the algorithm the Hessian matrix.

In our examples, we use the SCG implementation written by Nabney (2002) which uses the Polak and Ribière (1969) formulae to update the search direction at every iteration. The only inputs required for the optimization are the likelihood function and its analytical derivatives with respect to $\mathbf{X}$ and $\boldsymbol{\theta}$. The latter are provided in Appendix A.

4.3. Further considerations

An alternative solution to the Empirical Bayes estimate can be found by using a Markov Chain Monte Carlo algorithm. The interested reader should refer to Shi and Choi (2011, Section 8.2) where full implementation details are given.

Numerically, irrespective of whether a full or an empirical Bayes approach is used to obtain estimates $\widehat{\mathbf{X}}$ and $\widehat{\boldsymbol{\theta}}$, the inverse of the covariance matrix, $\mathbf{K}_y^{-1}$, is involved in Eq. (9). The cost of the log-likelihood evaluations is, hence, of order $O(N^3)$, where N is the sample size. As N increases, model training will slow down as the cost of the calculation becomes more and more prohibitive.

In those cases where the nominal data set is substantially large, training of the GPLV model can be sped up by selecting a subset $\mathcal{I}$ of size m, with $m \ll N$, from the original data set $\mathcal{D}$. Let us denote the remaining (unselected observations) as $\mathcal{J}$. By replacing $\mathcal{D}$ with $\mathcal{I}$, computational efficiencies are gained as the cost of the likelihood calculation will be of order $O(m^3)$ rather than $O(N^3)$. $\mathcal{I}$ is normally referred to as the *active set* and, obviously, its selection causes a reduction in the information available for inference (Shi and Choi, 2011, Section 3.3). What it is expected is that, if a good subset selection is made, most of the information will be kept. There are several criteria that can be used to partition $\mathcal{D}$ into $\mathcal{I}$ and $\mathcal{J}$. The most popular ones are probably based on the Kullback-Leibler divergence criterion and the process entropy. The latter criterion is used by the *Informative Vector Machine*, IVM, algorithm (Lawrence et al., 2003) which sequentially selects the points in $\mathcal{I}$ according to the reduction in the process' entropy that they cause. An IVM implementation of the GPLV model can be found in Lawrence (2005).

4.4. GPLV model prediction

The GPLV model prediction for a new but known input vector $\boldsymbol{x}^*$ is an extension of Eq. (5) to every output variable $\boldsymbol{y}_{(d)}$. Let us define $f_M(\boldsymbol{x}^*) = (f_1(\boldsymbol{x}^*), f_2(\boldsymbol{x}^*), \ldots, f_D(\boldsymbol{x}^*))^\intercal$. The joint distribution of the new enlarged matrix of outputs $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N, f_M(\boldsymbol{x}^*))^\intercal$ will still be multivariate normal; the prediction, $\widehat{\boldsymbol{y}}^*$, of $f_M(\boldsymbol{x}^*)|\mathcal{D}$ is also a multivariate normal distribution whose mean and common variance are given as

$$\begin{aligned} E(f_M(\boldsymbol{x}^*)|\mathcal{D}) &= \mathbf{Y}^\intercal \mathbf{K}_y^{-1} \boldsymbol{k}^* \\ \operatorname{var}(f_M(\boldsymbol{x}^*)|\mathcal{D}) &= \left(k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}^{*\intercal} \mathbf{K}_y^{-1} \boldsymbol{k}^*\right) \mathbf{I}_D \end{aligned} \tag{10}$$

where, as before, $\boldsymbol{k}^* = (k(\boldsymbol{x}^*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}^*, \boldsymbol{x}_n))^\intercal$ is the vector of covariances between the new input point, $\boldsymbol{x}^*$, and the training data $\boldsymbol{x}_i$, $i = 1, \ldots, N$; $\mathbf{I}_D$ is the D-dimensional identity matrix.

5. Projecting new observations onto the latent space

Given a training (nominal) set of D-dimensional observations $\mathbf{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)^\intercal$, their representation in the latent space can be found by maximizing Eq. (9). In other words, both the latent variables $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^\intercal$ and $\boldsymbol{\theta}$, the $\mathcal{GP}$ hyperparameters, can be considered known once the optimization is

completed. The model prediction, $\widehat{\mathbf{Y}}$, can then easily be found by applying Eq. (10).

Let us now say that a new observation $\boldsymbol{y}_j = (y_{j1}, \ldots, y_{jD})^\intercal$ becomes available. The problem of projecting that observation onto the latent space is concerned with finding $\boldsymbol{x}_j$, its associated latent variable representation. We provide two possible ways of doing so.

5.1. Method 1: GPLVM projection

Eq. (5) is a standard result from $\mathcal{GP}$s. For clarity, it can also be expressed as

$$\boldsymbol{y}_j | \boldsymbol{x}_j; \mathbf{X}, \boldsymbol{\theta} \sim \mathcal{N}_D(\widehat{\boldsymbol{y}}_j, s_j^2 \mathbf{I}_D) \tag{11}$$

where

$$\begin{aligned} \widehat{\boldsymbol{y}}_j &= \mathbf{Y}^\intercal \mathbf{K}_y^{-1} \boldsymbol{k}_j \\ s_j^2 &= k(\boldsymbol{x}_j, \boldsymbol{x}_j) - \boldsymbol{k}_j^\intercal \mathbf{K}_y^{-1} \boldsymbol{k}_j + \sigma_j^2 \end{aligned} \tag{12}$$

and $\boldsymbol{k}_j = (k(\boldsymbol{x}_j, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_j, \boldsymbol{x}_n))^\intercal$. Note that, as we observe $\boldsymbol{y}_j$ and not $f(\boldsymbol{x}_j)$, the uncertainty is higher and reflected via σ_j^2.

The only unknown parameters in Eq. (11) is $\boldsymbol{x}_j$. Its log-likelihood can be written as

$$\begin{aligned} \ell(\boldsymbol{x}_j; \boldsymbol{y}_j, \mathbf{X}, \boldsymbol{\theta}) = &- \frac{D}{2} \log(2\pi) - \frac{D}{2} \log(s_j^2) \\ &- \frac{1}{2(s_j^2)} (\boldsymbol{y}_j - \widehat{\boldsymbol{y}}_j)^\intercal (\boldsymbol{y}_j - \widehat{\boldsymbol{y}}_j) \end{aligned} \tag{13}$$

Additionally, by giving a Gaussian prior distribution to the latent variable $\boldsymbol{x}_j$, i.e. $\boldsymbol{x}_j \sim \mathcal{N}(0, \mathbf{I}_Q)$, then

$$p(\boldsymbol{x}_j) \propto \exp\left(-\frac{1}{2} \boldsymbol{x}_j^\intercal \boldsymbol{x}_j\right)$$

The MAP can therefore be found by maximizing the following log-likelihood function

$$\ell_{MAP}(\boldsymbol{x}_j; \boldsymbol{y}_j, \mathbf{X}, \boldsymbol{\theta}) = \ell(\boldsymbol{x}_j; \boldsymbol{y}_j, \mathbf{X}, \boldsymbol{\theta}) - \frac{1}{2} \boldsymbol{x}_j^\intercal \boldsymbol{x}_j \tag{14}$$

where constant terms have been omitted.

The same *scaled conjugate gradient* non-linear optimizer, used to fit the GPLV model in the first instance, can be employed to determine $\boldsymbol{x}_j$; now the objective function to maximize is given by Eq. (14) and the gradients thereof with respect to $\boldsymbol{x}_j$ are shown in Appendix B.

This is the method proposed by Ge and Song (2010) to deal with new observations. Potential users must be very cautions, however, as the objective function given by Eq. (14) is non-convex. A procedure must be put in place to make sure that the global maximum is chosen when projecting every new observation. While this is relatively simple when the underlying dimensionality of the latent space is low, the problem is far from trivial when this is not the case. We show a simple simulation example where this problem will be further explained.

5.2. Method 2: Neural Network projection

The procedure we prefer to follow in order to use the GPLVM for process monitoring is to build two neural network models following the idea introduced by Dong and McAvoy (1996). By doing this we avoid dealing with the non-convexity problem altogether. The first ANN would map the D-dimensional data space onto its underlying Q-dimensional latent space. The second ANN would then map the Q-dimensional latent variables onto the GPLVM model prediction as shown in Figure 1.

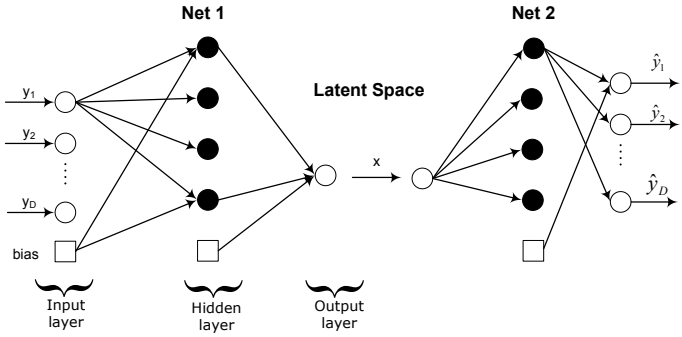

Figure 1: Architecture of the neural networks needed for process monitoring; only 1 latent variable.

Both networks have one hidden layer in which the inputs and outputs are fully determined and known. The only remaining unknown is M, the number of hidden units, that must be adjusted to give the best predictive performance. M, in turn, controls the total number of network parameters (model complexity) so we can expect an optimum value to exist giving the best generalization performance.

Bishop (2006, Section 5.5) cites different procedures that could be used to for this purpose. The method we have followed to control network complexity is early *early-stopping*. The available data is divided into three subsets. The first subset is the *training set*, used to compute gradients and the network parameters. The second subset is the *validation set*

whose error is monitored during the training process. The training set error is a non-increasing function of the iteration index. On the other hand, the validation set error normally decreases during the initial phase of training; however, as the network begins to overfit the training data, the error of the validation data set will typically begin to rise. When this latter error increases during six consecutive iterations, training is stopped and the network parameters at the minimum of the validation error are adopted. The third subset is the *test set*, which it is only used to assess the generalization performance of the network.

6. Online monitoring strategy

The GPLVM-based monitoring method is similar to that using linear PCA in the sense that the same monitoring statistics are used. Monitoring can be made on the Hotelling's T^2 statistic, the Q-statistic (Squared Prediction Error) or directly on the latent variables themselves. Although the $\mathcal{GP}$ latent variables will be representative of the underlying dimensionality of the system, they lack physical interpretation which simply makes the problem of fault diagnosis more complicated. Serradilla and Shi (2010) offer details portraying the GPLV model as the building block of a bigger class of models denoted as *Gaussian process factor analysis models*, GPFA. Their main advantage is that some physical interpretation can be given to the latent variables if the model structure is carefully designed.

In terms of the process faults, there two kind of process abnormalities that can develop in a chemical system, Zhang et al. (1996). Firstly, the relationship between the process variables could change. What it is expected in this situation is that the difference between the original observations $\boldsymbol{y}_{(d)}$ and the model prediction $\hat{\boldsymbol{y}}_{(d)}$ be large. These faults can be detected by monitoring the *Squared Prediction Error*, SPE. And secondly, the basic relationship between the process variables could remain unchanged but the process variables could present a variability higher than those in the nominal data. This abnormality would be observable if we were to monitor the latent variables directly. These faults can be detected by using Hotelling's T^2 Statistic.

6.1. latent scores monitoring

The latent variables have been constraint to be $\boldsymbol{x}_i \sim \mathcal{N}(0, \mathbf{I}_Q)$, i.e. uncorrelated (independent) and normally distributed. Therefore they could be monitor directly; confidence limits can be built from the standard normal distribution.

6.2. Squared Prediction Error

The SPE is simply a measure of the lack of fit in the *observational space*. It is given by

$$SPE_i = \boldsymbol{e}_i^\intercal \boldsymbol{e}_i = (\boldsymbol{y}_i - \widehat{\boldsymbol{y}}_i)^\intercal (\boldsymbol{y}_i - \widehat{\boldsymbol{y}}_i) \tag{15}$$

where, for observations in the nominal data set $(i = 1, \ldots, N)$, $\widehat{\boldsymbol{y}}_i$ is given by Eq. (10) whereas, for new observations, it will be the output from the second neural network.

Confidence limits for the SPE can be obtained by fitting a weighted χ^2-distribution to the squared errors generated from normal operating condition data as explained by Nomikos and MacGregor (1995).

6.3. Hotelling's statistic

This statistic is a measure of the variation of each sample within the latent variable space. Therefore and unlike the SPE, monitoring of the Hotelling's T^2-statistic occurs directly in the latent space. It can be found as

$$\mathrm{T}_i^2 = \boldsymbol{x}_i^\intercal \boldsymbol{\Lambda}_Q^{-1} \boldsymbol{x}_i = \sum_{q=1}^{Q} \frac{x_{iq}^2}{\lambda_q} \tag{16}$$

where $\boldsymbol{\Lambda}_Q$ is *Q-diagonal* matrix with elements λ_q being the variance of each one of the latent variables. For observations in the nominal data set $(i = 1, \ldots, N)$, $\boldsymbol{x}_i$ is the output from fitting the GPLV model whereas, for new observations, it will be the output from the first neural network.

Confidence limits for the Hotelling's T^2-statistic can be obtained using the empirical reference distribution of the training data or the F-distribution, as follows

$$T^2_{Q,N;\alpha} = \frac{Q\,(N-1)}{N-Q} F_{Q,N-Q;\alpha}$$

where α is the significance level.

6.4. Monitoring scheme

The monitoring strategy can be summarized as follows

I. **Nominal model**

1. Select the nominal data $\mathbf{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)^\intercal$ from observations where the process is known to be behaving as intended.

2. Fix the number of latent variables Q, for instance, by setting a desired *percentage of the variance explained*.
3. Build the GPLV model. The outputs from this model will be the latent variables, $\mathbf{X}$, as well as the $\mathcal{GP}$ hyperparameters, $\boldsymbol{\theta}$.
4. Use the fitted model to find the confidence limits for the SPE and the T^2 statistics.

II. **New Observations**

1. Method 1. As recently proposed by Ge and Song (2010); this is not advisable unless our algorithm is able to find the maximum of a non-convex function. Failing to do so will increase the false alarm rate.
2. Method 2. This requires the construction of two auxiliary ANNs models. The mappings are as follows

$$\textbf{Net-1: } \mathbf{Y} \in \mathcal{R}^D \mapsto \mathbf{X} \in \mathcal{R}^Q$$
$$\textbf{Net-2: } \mathbf{X} \in \mathcal{R}^Q \mapsto \widehat{\mathbf{Y}} \in \mathcal{R}^D$$

3. Calculate SPE_j, T_j^2 for every new observation j or monitor the latent variables directly.

7. Simulation

7.1. A simple example

This first example refers to the system presented by Dong and McAvoy (1996). There are three variables, $D = 3$, but only one underlying latent variable, $Q = 1$.

Data generation

The data is generated following these steps:

(a) Assume that $x \sim \mathcal{U}(1.01, 2)$ and e_d $(d = 1, 2, 3) \sim \mathcal{N}(0, 0.01^2)$ is random independent noise.

(b) Generate 100 observations of *normal operating data* (y_1, y_2, y_3) following the equations on the left panel. Then simulate a fault in y_3 by generating 100 observations following the relationships on the right hand panel

$$\begin{aligned} y_1 &= x + e_1 & y_1 &= x + e_1 \\ y_2 &= x^2 - 3x + e_2 & y_2 &= x^2 - 3x + e_2 \\ y_3 &= -x^3 + 3x^2 + e_3 & y_3 &= -1.1x^3 + 3.2x^2 + e_3 \end{aligned} \tag{17}$$

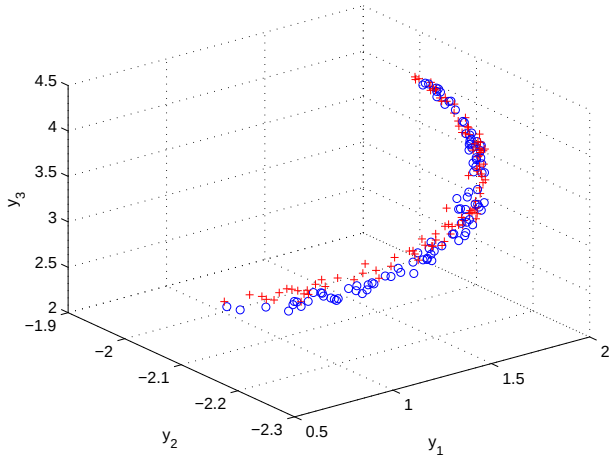

Figure 2: Data set for normal condition (o) and fault condition (+)

Figure 2 shows the data sets for the normal condition and fault condition; notice how the faulty observations detach from the nominal data in the y_3 direction.

Model training

The objective is to model the *nominal data* in the previous system by using the non-linear GPLV model defined in Section (4). Latent positions were initialized using linear PCA while the $\mathcal{GP}$ hyperparameters are given random positive values. The prediction for the training data, as given by Eq. (10), is shown in Fig 3. As it can be the seen the three *iid* $\mathcal{GP}$s do provide an excellent and smooth approximation to the data.

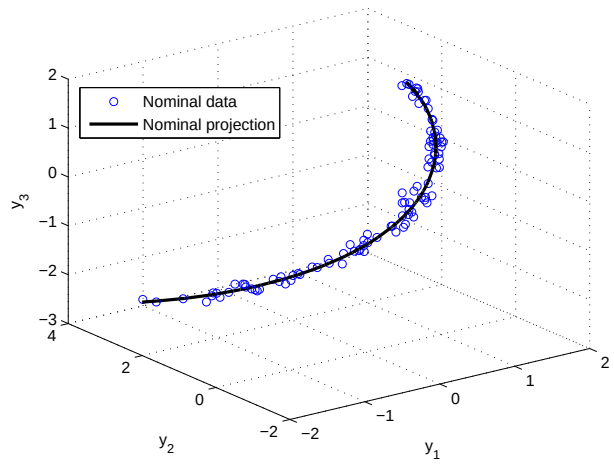

Figure 3: Normalized *nominal data* and the GPLV model prediction

One of the advantages of using this simulation is that the generating latent variable, $\boldsymbol{x} \in \mathcal{R}^N$, is fully known. It can therefore be compared with its estimate, $\widehat{\boldsymbol{x}} \in \mathcal{R}^N$, obtained by fitting the GPLV model. The standardized values of both variables are shown in Figure 4. The correlation coefficient is $\text{cor}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = 0.999$, showing the suitability of the proposed model for this non-linear system.

Another 'goodness of fit' measure commonly reported is the percentage of variance in the original data explained. The single latent variable obtained with the GPLV model can explain 99.6% of that variance. Fitting a linear model to nonlinear data is not entirely appropriate; however, for comparison purposes, if we used one linear principal component we would be able to explain 66.1% of the original variance, thus highlighting the improvement achieved with the non-parametric model.

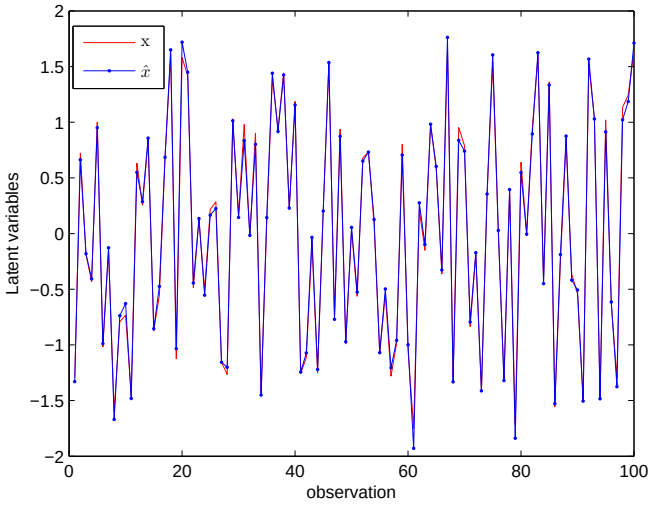

Figure 4: $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$ for the nominal data

New observations: GPLV model projection

We have generated 100 samples with a known fault in y_3. Before projecting every observation onto the latent space, let us focus on any single one of them, which we denote as $\boldsymbol{y}_j$. The aim of the projection is to determine the latent variable x_j associated with the available observation. As previously explained, this can be done by maximizing Eq. (14) with respect to x_j. In this case, as the the latent space is monodimensional, the log-likelihood can be visualized for different values of x_j, Figure 5, upper left-hand corner. What this figure highlights is that the objective function is not convex and, in this particular case, three maxima occur. They are shown amplified in the remaining three panels of the figure.

The projection achieved will be different depending on the starting point chosen to start the iterative algorithm. In this case, there is no complication in making the algorithm converge to the global maximum; we simply choose several random starting points and select the one with the highest maximum. However, for multivariate optimization problems, we will not be able to guarantee that the global maximum has been achieved at all times.

With this caveat in mind, all the faulty observations can be projected onto the latent space. The model predicted values, $\widehat{\boldsymbol{y}}_j$, can be obtain by using Eq. (10). This then allows the calculation of the squared prediction error.

The SPE for both the nominal data and the faulty observations is shown in Fig. 6 along with 95% and 99% confidence intervals calculated as described previously. As a measure of fault detection performance we can report the number of observations outside the 99% confidence limit. Using this model, the SPE for only one observation is outside this limit in the nominal data set. On the other hand, the method is able to pick up 9 observations in the 100 samples simulated for the faulty data set.

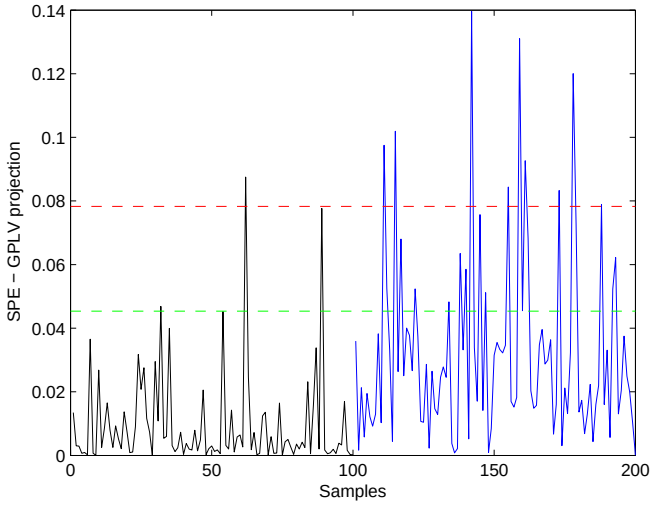

Figure 6: SPE for nominal and faulty observations using the GPLV model projection method.

For comparison purposes, a linear PCA model with only one latent variable has also been built. The SPE for this model is shown in Figure 7. There are 2 and 4 observations outside the 99% confidence lim-

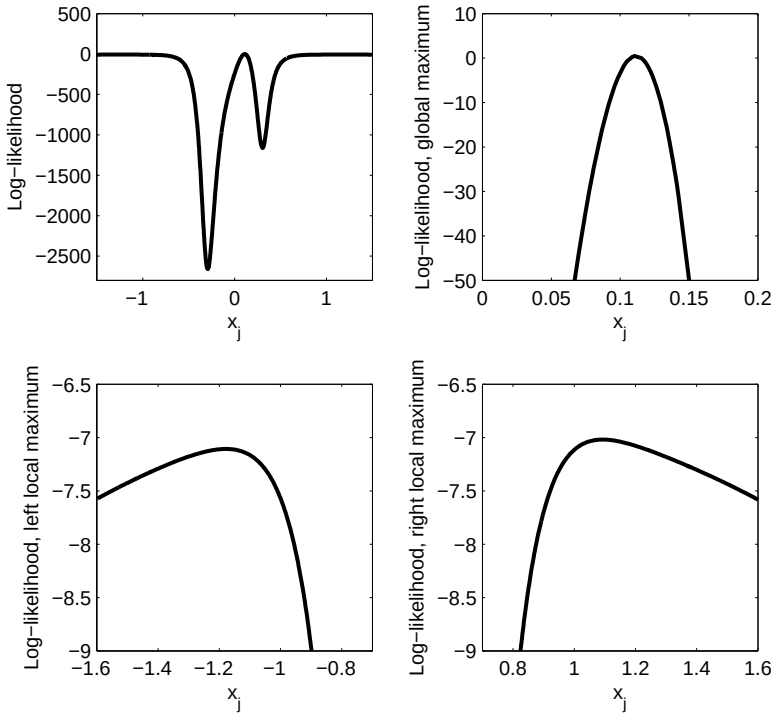

Figure 5: Log-likelihood (projection of a *faulty observation*) and amplified areas where the 3 maxima occur. Global maximum is at $x_j = 0.1115$, left-side local maximum at $x_j = -1.1781$ and right-side local maximum at $x_j = 1.0928$

its respectively for the nominal and faulty data sets. Therefore, as expected, the model is unable to detect the faulty condition; to do so would require of an additional latent variable (not shown).

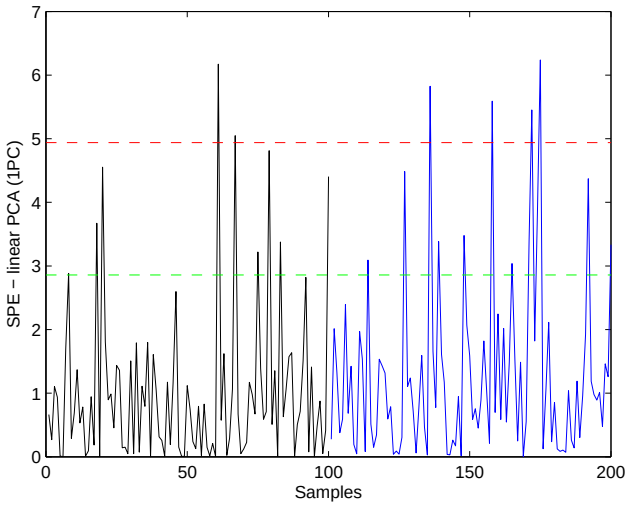

Figure 7: SPE for nominal and faulty observations using 1 linear principal component.

New observations: ANN projection

Two networks were trained using the nominal observations with the *scaled conjugate gradient* backpropagation algorithm. As previously mentioned, network complexity was controlled by using *early stopping*. The squared prediction error for both the nominal data and the faulty observations is shown in Fig. 8 along with 95% and 99% confidence intervals. In this case, there are 23 observations outside the 99% confidence limits. Note that this performance is similar to the that reported by Dong and McAvoy (1996), where the authors argue that their method using principal curves was able to detect 26 observations outside this limit. Note also that, for this particular simple simulation, it seems that projecting new observations using NN is more efficient that doing so with the GPLV model directly.

7.2. Simulated CSTR process

In this example, a non-isothermal continuous stirred tank reactor (CSTR) is simulated. This example has been widely used in the literature to test other non-parametric methods; see for instance Choi et al. (2005), Choi et al. (2008) and Alcala and Qin (2010).

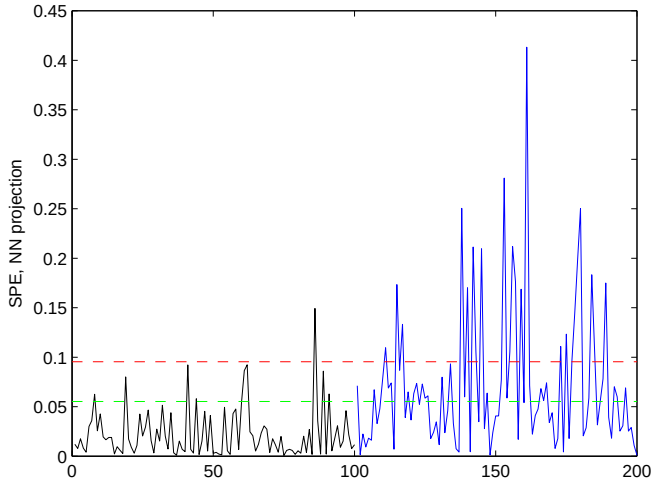

Figure 8: SPE for nominal and faulty observations using the NN projection method.

Process description

The process flow is depicted in Figure 9. The reaction, $A \rightarrow B$, is irreversible, exothermic and takes place in liquid phase. A feed stream of reactant A with flow rate F_a is premixed with a solvent stream flowing at a rate F_s; the concentration of reactant A in both streams is C_{aa} and C_{as} respectively. This premixed stream, with reactant concentration C_i and flow rate F, is then fed into the jacketed reactor where the reaction takes place.

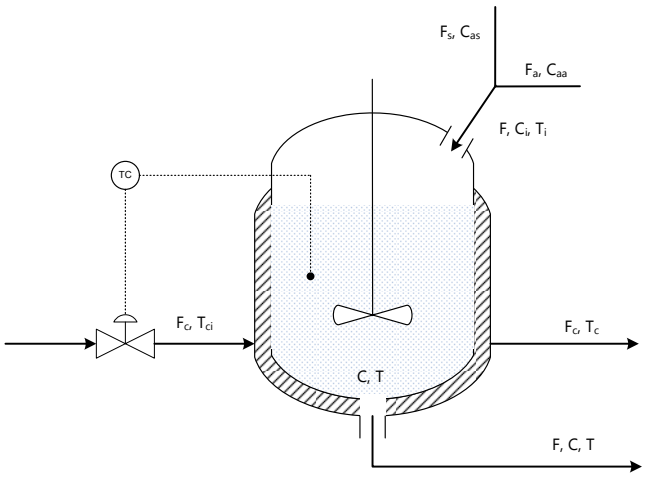

Figure 9: Process flow diagram of the non-isothermal CSTR system.

The system has only a PI control loop whose aim is to maintain the outlet temperature T at a set value. This is done by controlling the flow of cooling water, F_c, which enters the reactor jacket at a temperature T_{ci} and leaves at a temperature T_c. The model assumes perfect mixing, constant physical properties and negligible shaft work. The dynamic behaviour of this process is governed by two ordinary differential equations(ODE). Firstly, the mass balance for reactant A

$$V\frac{dC}{dt} = F(C - C_i) - Vr \tag{18}$$

where V is the volume of reacting liquid; r is an Arrhenius-type reaction rate given as $r = k_0 e^{-E/RT} C$ with k_0 being the pre-exponential factor and R the gas constant. The second ODE is the global energy balance written as follows:

$$V\rho c_p \frac{dT}{dt} = \rho c_p F(T_i - T) - \frac{aF_c^{b+1}}{F_c + F_c^b/2\rho_c c_{pc}}(T - T_{ci}) + (-\Delta H_r)Vr \tag{19}$$

where ρ and ρ_c are the densities of the reacting mixture and the cooling water, respectively, whereas c_p and c_{pc} as their specific heat capacities; ΔH_r is the heat of the reaction. A summary of the process variables and simulation parameters is given in Table 1.

Table 1: CSTR process variables and parameters summary

Variable type	
Controlled variable:	T
Manipulated variable:	F_c
Disturbances:	$C_{aa}, C_{as}, F_s, T_i, T_{ci}, a_1, a_2$
Measured variables:	$T_{ci}, T_i, C_{aa}, C_{as}, F_s, F_c, C, T$
Parameters	
$V = 1\text{m}^3; \rho = 10^6\text{g/m}^3; \rho_c = 10^6\text{g/m}^3;$	
$c_p = 1\text{cal/(a} \cdot \text{K)}; c_{pc} = 1\text{cal/(a} \cdot \text{K)};$	
$k_0 = 10^{10}\text{min}^{-1}; b = 0.5;$	
$a = 1.678 \cdot 10^6\text{cal/min}; \Delta H_r = -1.3 \cdot 10^7\text{cal/kmol}$	

All process disturbances are simulated as first order autoregressive processes; variables a_1 and a_2 in Table 1 are used to simulate degradation in the reaction rate due to impurities and fouling of the water-cooled heat exchanger respectively. Likewise, process noise is added to all measured variables. Further details about initial conditions, controller information and disturbances simulation are given by Yoon and MacGregor (2001).

Complex fault generation

Yoon and MacGregor (2001) categorize abnormal operating conditions as either *simple* or *complex* faults; in the former case, a fault occurring in one variable does not propagate into other variables whereas in the latter situation, the effect of the fault is seen by other process variables. To clarify this, let us generate 100 observations and introduce a complex fault at $t = 50$ minutes; the fault is simply a bias of 1°C in the outlet temperature sensor. A time series plot of both T and F_c is given in Figure 10. Note that as the outlet temperature is the controlled variable, the feedback controller will act to remove this bias at the expense of increasing the cooling water flow rate.

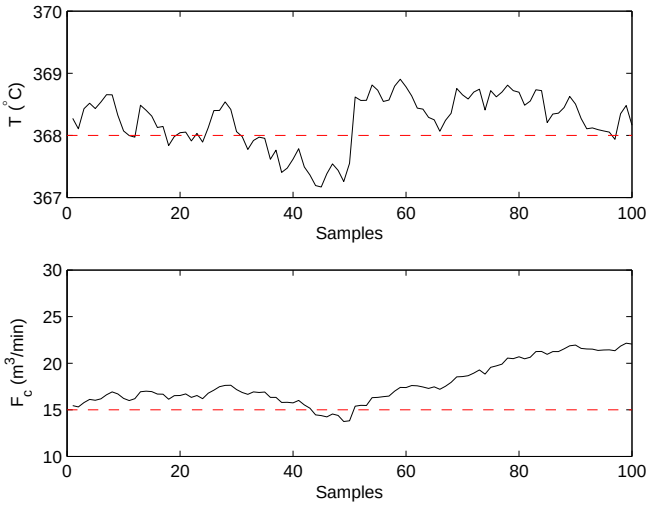

Figure 10: Bias fault of 1°C in the outlet temperature sensor occurring at $t = 50$ min.; controller set point at 368°C.

Complex fault detection

The training data is obtained by simulating the CSTR process for 200 minutes. A further 100 observations are generated containing the 1°C permanent bias in the outlet temperature sensor.

First, linear PCA models have been built; the percentage of variance explained as a function of the number of principal components kept in the model is given in Table 2. Based on these results and the amount of noise in the data, up to six principal components, which explain 96.6% of the original variance, would need to be kept in order to have a good model.

Two GPLV models have also been built with the 200 hundred observations from the training data, $\mathbf{Y}$. By fitting the GPLV model, both an estimation of the underlying latent variables, $\widehat{\mathbf{X}}$, and the $\mathcal{GP}$ parameters, $\boldsymbol{\theta}$, are found; this then allows to determine the GPLV model projection, $\widehat{\mathbf{Y}}$, as given by Eq. (10). To monitor the process, we have then built two neural network models. The first network builds the map from $\mathbf{Y} \mapsto \widehat{\mathbf{X}}$ while the second network takes back the observations from the latent space into their original dimensionality, i.e. $\widehat{\mathbf{X}} \mapsto \widehat{\mathbf{Y}}$.

Table 2: Results for PCA and the GPLV model

Latent variables	Explained variance (%)	
	PCA	GPLV model
1	35.6	84.8
2	55.3	96.0
3	71.4	-
4	83.7	-
5	90.5	-
6	96.6	-
7	99.3	-
8	100.0	-

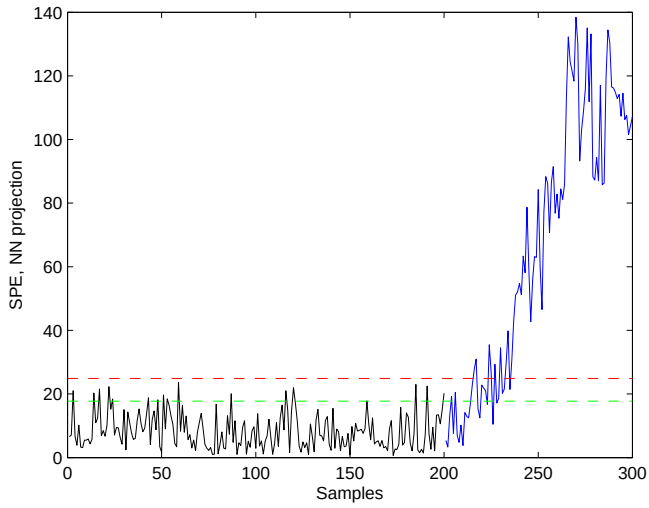

Figure 11: Bias fault of 1°C in the sensor for outlet temperature. Only one latent variable.

Let us first consider the case with only one underlying dimension. As shown in Table 2, this latent variable is able to account for around 85% of the original variance. The SPE for this example is given in Figure 11 along with the 95% and 99% confidence limits. As it can be seen, shortly after the sample 200, the SPE starts moving pretty abruptly outside both limits as a result of the bias fault being introduced.

A second GPLV model with two latent variables

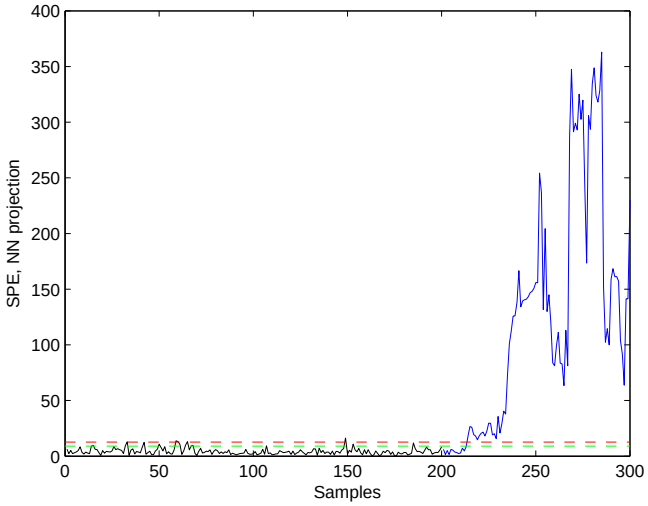

Figure 12: Bias fault of 1°C in the sensor for outlet temperature. Two latent variables.

has also been built; one could argue that the true dimensionality of this process is two, as the two latent variables are able to explain 96% of the variation in the original data. As before, the SPE has been calculated and plotted in Figure 12. It is very obvious, even by a visual comparison, that this latter model is far more sensitive that the model with only one latent variable. Not only the magnitude of the SPE for the training data reduces as a result of having an improved model but also the range of faulty data SPE increases quite dramatically.

8. Conclusions

The $\mathcal{GP}$ latent variable methodology has already been introduced in the field of *multivariate statistical process control* by Ge and Song (2010). When projecting new observations into the GPLV space, however, their approach is prone to the potentially serious pitfall of having to determine the global maximum of a likelihood function which is not convex. As the dimensionality of the latent space becomes larger, that problem becomes less and less trivial.

This paper offers a review of similar non-parametric methods available in the literature. It provides a detailed description of how to determine the latent space parameters by using the *Empirical Bayes estimation* method; it also provides references to alternative estimation procedures available in the literature. To deal with the non-convexity problem of the likelihood function when projecting new observations into the latent space, we propose the use of two neural network models; this is in line with previous approaches that have been successfully applied. By using a simple simulation example we show how the proposed GPLV model can unravel complicated non-linear relationships and find the underlying latent variables driving the process. We have also successfully demonstrated the method applicability with data from highly non-linear CSTR. We expect that this paper provides enough tools to facilitate the use of the GPLV model and can only foresee this methodology to keep growing as further applications are found.

Acknowledgements

J. Serradilla would like to acknowledge the financial support throughout his PhD study from the EPSRC (EP/P502624/1) and BP Oil International Ltd (GPTL/MGA/50305/3655). Additionally, thanks to S. W. Choi for providing the CSTR model to test the method.

Appendix A. GPLVM derivatives

Training of the GPLV model requires the maximization of the log-likelihood function given by Eq. (9). The analytical derivatives of this function with respect to the latent positions, $\mathbf{X}$, and the $\mathcal{GP}$ hyper-parameters, $\boldsymbol{\theta}$, are also needed for the SCG optimizer. Note that we refer to every element of $\boldsymbol{\theta}$ as θ_j.

These gradients can be calculated using the chain rule as follows

$$\begin{aligned}\frac{\partial \ell(\mathbf{X},\boldsymbol{\theta};\mathbf{Y})}{\partial x_{iq}} &= \mathrm{tr}\left[\left(\frac{\partial \ell}{\partial \mathbf{K}_y}\right)^{\intercal}\left(\frac{\partial \mathbf{K}_y}{\partial x_{iq}}\right)\right]\\ \frac{\partial \ell(\mathbf{X},\boldsymbol{\theta};\mathbf{Y})}{\partial \theta_j} &= \mathrm{tr}\left[\left(\frac{\partial \ell}{\partial \mathbf{K}_y}\right)^{\intercal}\left(\frac{\partial \mathbf{K}_y}{\partial \theta_j}\right)\right]\end{aligned} \tag{A.1}$$

The common derivative, i.e. the $N \times N$ gradient of the log-likelihood with respect to the kernel matrix, is independent of the chosen covariance function and is given by

$$\left(\frac{\partial \ell}{\partial \mathbf{K}_y}\right) = -\frac{D}{2}\mathbf{K}_y^{-1} + \frac{1}{2}\mathbf{K}_y^{-1}\mathbf{Y}\mathbf{Y}^{\intercal}\mathbf{K}_y^{-1} \tag{A.2}$$

Hyper-parameters gradient

The kernel matrix $\mathbf{K}_y$ is a function of $\boldsymbol{\theta}$ as shown by Eq. (4). Let us now rewrite the covariance function, Eq. (2), as

$$\begin{aligned}k(\boldsymbol{x}_i,\boldsymbol{x}_j;\boldsymbol{\theta}) &= \nu_0 \exp\left\{-\frac{1}{2}\sum_{q=1}^{Q}\gamma(x_{iq}-x_{jq})^2\right\}\\ &= \nu_0 \exp\left\{-\frac{1}{2}\gamma d_{ij}^2\right\}\end{aligned} \tag{A.3}$$

where $d_{ij}^2 = \sum_{q=1}^{Q}(x_{iq}-x_{jq})^2$ is simply the squared euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Let us also define $\mathbf{D} = \left(d_{ij}^2\right)$, i.e. the $N \times N$ matrix of squared euclidean distances.

Every $\left(\frac{\partial \mathbf{K}_y}{\partial \theta_j}\right)$ is an $N \times N$ matrix given as follows:

$$\begin{aligned}\left(\frac{\partial \mathbf{K}_y}{\partial \nu_0}\right) &= \frac{1}{\nu_0}\mathbf{K}\\ \left(\frac{\partial \mathbf{K}_y}{\partial \gamma}\right) &= \left(-\frac{1}{2}\right)\mathbf{D}\odot\mathbf{K}\\ \left(\frac{\partial \mathbf{K}_y}{\partial \sigma^2}\right) &= \mathbf{I}\end{aligned} \tag{A.4}$$

where $\mathbf{I}$ is the N-dimensional identity matrix and $\odot$ represents the Hadamard (element-wise) product.

A further constraint in the GPLV model is that all hyper-parameters must be positive. In order to achieve that, it is better to reparametrize and carry out the optimization in the log-space. That is easily achieved combining the following equality

$$\left(\frac{\partial \mathbf{K}_y}{\partial \log(\theta_j)}\right) = \left(\frac{\partial \mathbf{K}_y}{\partial \theta_j}\right)\theta_j$$

with equation Eq. (A.1).

Latent positions gradient

Finally $\left(\frac{\partial \mathbf{K}_y}{\partial x_{iq}}\right)$ is a $N \times N$ symmetric matrix of all zeros but the i^{th} row/column. The elements of this row/column are given by

$$\left(\frac{\partial \mathbf{K}_y}{\partial x_{iq}}\right) = -\gamma\begin{pmatrix}(x_{iq}-x_{1q})k(\boldsymbol{x}_1,\boldsymbol{x}_i)\\(x_{iq}-x_{2q})k(\boldsymbol{x}_2,\boldsymbol{x}_i)\\\vdots\\(x_{iq}-x_{Nq})k(\boldsymbol{x}_N,\boldsymbol{x}_i)\end{pmatrix}_i$$

where, notationally, the subscript i in the right hand-side of the equation is included to refer only to the elements in the i^{th} row/column of the gradient matrix.

Furthermore, note the extra term in Eq. (9) which is independent of the kernel matrix, $\frac{1}{2}\mathrm{tr}(\mathbf{X}\mathbf{X}^{\intercal})$. As $\left(\frac{\partial}{\partial \mathbf{X}}\mathrm{tr}(\mathbf{X}^{\intercal}\mathbf{X})\right) = 2\mathbf{X}$ it finally follows that

$$\left(\frac{\partial \ell(\mathbf{X},\boldsymbol{\theta};\mathbf{Y})}{\partial \mathbf{X}}\right)_{MAP} = \left(\frac{\partial \ell(\mathbf{X},\boldsymbol{\theta};\mathbf{Y})}{\partial \mathbf{X}}\right) - \mathbf{X} \tag{A.5}$$

Appendix B. GPLVM projection gradients

The first derivatives of the log-likelihood, Eq. (13), with respect the new latent variables can be found by applying the chain rule as

$$\frac{\partial \ell(\boldsymbol{x}_j;\boldsymbol{y}_j,\mathbf{X},\boldsymbol{\theta})}{\partial x_{jq}} = \left[\left(\frac{\partial \ell}{\partial \boldsymbol{k}_j}\right)^{\intercal}\left(\frac{\partial \boldsymbol{k}_j}{\partial x_{jq}}\right)\right] \tag{B.1}$$

Let us first re-express the log-likelihood as

$$\begin{aligned}\ell(\boldsymbol{x}_j;\boldsymbol{y}_j,\mathbf{X},\boldsymbol{\theta}) &= -\frac{D}{2}\log(s_j^2)\\ &\quad -\frac{1}{2(s_j^2)}(\boldsymbol{y}_j-\widehat{\boldsymbol{y}}_j)^\intercal(\boldsymbol{y}_j-\widehat{\boldsymbol{y}}_j)\\ &= -\frac{D}{2}\log(s_j^2)-\frac{1}{2(s_j^2)}\boldsymbol{e}_j^\intercal\boldsymbol{e}_j\end{aligned}$$

where $\boldsymbol{e}_j = \boldsymbol{y}_j - \widehat{\boldsymbol{y}}_j$. Therefore

$$\begin{aligned}\left(\frac{\partial\ell}{\partial\boldsymbol{k}_j}\right) &= \left(\frac{D}{2}\right)\frac{1}{(s_j^2)}\left(2\mathbf{K}^{-1}\boldsymbol{k}_j\right)\\ &\quad -\frac{1}{2(s_j^2)^2}\left[-2\mathbf{K}^{-1}\mathbf{Y}\boldsymbol{e}_j(s_j^2)-\boldsymbol{e}_j^\intercal\boldsymbol{e}_j(-2\mathbf{K}^{-1}\boldsymbol{k}_j)\right]\\ &= \frac{D\mathbf{K}^{-1}\boldsymbol{k}_j}{s_j^2}+\frac{\mathbf{K}^{-1}\mathbf{Y}\boldsymbol{e}_j}{s_j^2}-\frac{\boldsymbol{e}_j^\intercal\boldsymbol{e}_j\mathbf{K}^{-1}\boldsymbol{k}_j}{(s_j^2)^2}\end{aligned}$$

and

$$\left(\frac{\partial\ell}{\partial\boldsymbol{k}_j}\right)^\intercal = \frac{D\boldsymbol{k}_j^\intercal\mathbf{K}^{-1}+\boldsymbol{e}_j^\intercal\mathbf{Y}^\intercal\mathbf{K}^{-1}}{s_j^2}-\frac{\boldsymbol{e}_j^\intercal\boldsymbol{e}_j\boldsymbol{k}_j^\intercal\mathbf{K}^{-1}}{(s_j^2)^2}$$

Finally $\left(\frac{\partial\boldsymbol{k}_j}{\partial x_{jq}}\right)$ is the following $N\times 1$ vector

$$\left(\frac{\partial\boldsymbol{k}_j}{\partial x_{jq}}\right) = -\gamma\begin{pmatrix}(x_{jq}-x_{1q})k(\boldsymbol{x}_1,\boldsymbol{x}_j)\\(x_{jq}-x_{2q})k(\boldsymbol{x}_2,\boldsymbol{x}_j)\\\vdots\\(x_{jq}-x_{Nq})k(\boldsymbol{x}_N,\boldsymbol{x}_j)\end{pmatrix}$$

And, as $\frac{\partial}{\partial\boldsymbol{x}_j}\left(\frac{1}{2}\boldsymbol{x}_j^\intercal\boldsymbol{x}_j\right)=\boldsymbol{x}_j$, we finally have the gradient of Eq. (14) with respect to $\boldsymbol{x}_j$ as

$$\left(\frac{\partial\ell(\boldsymbol{x}_j;\boldsymbol{y}_j,\mathbf{X},\boldsymbol{\theta})}{\partial\boldsymbol{x}_j}\right)_{MAP} = \left(\frac{\partial\ell(\boldsymbol{x}_j;\boldsymbol{y}_j,\mathbf{X},\boldsymbol{\theta})}{\partial\boldsymbol{x}_j}\right)-\boldsymbol{x}_j$$

References

Alcala, C. F., Qin, S. J., 2010. Reconstruction-based contribution for process monitoring with kernel principal component analysis. Industrial & Engineering Chemistry Research 49 (17), 7849–7857.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer.

Choi, S. W., Lee, C., Lee, J.-M., Hyun Park, J., Lee, I.-B., 2005. Fault detection and identification of nonlinear processes based on kernel pca. Chemometrics and Intelligent Laboratory Systems 75, 55–67.

Choi, S. W., Morris, J., Lee, I.-B., 2008. Nonlinear multiscale modelling for fault detection and identification. Chemical Engineering Science 63, 2252–2266.

Dong, D., McAvoy, T., 1996. Nonlinear principal component analysis based on principal curves and neural networks. Computers and Chemical Engineering 20 (1), 65–78.

Ge, Z., Song, Z., 2010. Nonlinear probabilistic monitoring based on the gaussian process latent variable model. Industrial & Engineering Chemistry Research 49 (10), 4792–4799.

Jia, F., Martin, E. B., Morris, A. J., 1998. Non-linear principal components analysis for process fault detection. Computers & Chemical Engineering 22 (Supplement 1), S851 – S854, european Symposium on Computer Aided Process Engineering-8.

Kourti, T., 2002. Process analysis and abnormal situation detection: from theory to practice. Control Systems Magazine, IEEE 22 (5), 10 – 25.

Kourti, T., MacGregor, J., 1995. Process analysis, monitoring and diagnosis, using multivariate projection methods. Chemometrics and Intelligent Laboratory Systems 28 (1), 3 – 21.

Lawrence, N., 2005. Probabilistic non-linear principal component analysis with gaussian process latent variable models. Journal of Machine Learning Research 6, 1783–1816.

Lawrence, N. D., Seeger, M., Herbrich, R., 2003. Fast sparse gaussian process methods: The informative vector machine. In: Becker, S., Thrun, S., Obermayer, K. (Eds.), Advances in Neural Information Processing Systems. Vol. 15. MIT Press, Cambridge, MA, pp. 625–632.

McCabe, G. P., 1984. Principal variables. Technometrics 26 (2), 137–144.

Nabney, I., 2002. NETLAB: algorithms for pattern recognition. Advances in pattern recognition. Springer-Verlag, London, Berlin, Heidelberg.

Neal, R. M., 1994. Bayesian learning for neural networks. Ph.D. thesis, Dept. of Computer Science, University of Toronto.

Nocedal, J., Wright, S., 2006. Numerical Optimization, 2nd Edition. Springer.

Nomikos, P., MacGregor, J. F., 1995. Multivariate spc charts for monitoring batch processes. Technometrics 37 (1), 41–59.

Polak, E., Ribière, G., 1969. Note sur la convergence de méthodes de directions conjuguées. Revue française d'informatique et de recherche opérationnelle 3 (1), 35–43.

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

Serradilla, J., Shi, J. Q., 2010. Gaussian process factor analysis analysis model with applications in stochastic monitoring. Tech. rep., School of Maths & Stats, Newcasle University, UK.

Shi, J. Q., Choi, T., 2011. Gaussian Process Regression Analysis for Functional Data, 1st Edition. Chapman & Hall/CRC, London.

Simoglou, A., Martin, E. B., Morris, A. J., 2000. Multivariate statistical process control of an industrial fluidised-bed reactor. Control Engineering Practice 8 (8), 893–909.

Srinivasan, R., Qian, M., 2007. State-specific key variables for monitoring multi-state processes. Chemical Engineering Research and Design 85 (12), 1630–1644.

Tan, S., Mavrovouniotis, M. L., 1995. Reducing data dimensionality through optimizing neural network inputs. AIChE Journal 41, 1471 – 1480.

Tipping, M., Bishop, C., 1999. Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B 61, 611–622.

Wold, S., Esbensen, K., Geladi, P., 1987. Principal components

analysis. Chemometr. Intell. Lab. System 2, 37–52.
Yoon, S., MacGregor, J. F., 2001. Fault diagnosis with multivariate statistical models, part i: using steady state fault signatures. Journal of Process Control 11, 387–400.
Zhang, J., Martin, E., Morris, A., 1996. Fault detection and diagnosis using multivariate statistical techniques. Transactions of the Institution of Chemical Engineers 74, 89–96.