FAULT DETECTION USING RADIAL BASIS FUNCTION NETWORK AND POLYGONAL LINE

B. Bhushan, J. A. Romagnoli, D. Wang

Centre for Process Systems Engineering, Department of Chemical Engineering
The University of Sydney, NSW, 2006, Australia
Email: {bharat, jose, dwang} @chem.eng.usyd.edu.au

Abstract: This paper proposes a novel approach for the reduction of the dimensionality of non-linear data based on radial basis function (RBF) network and polygonal line (PL). A method is suggested to find out the optimum number of nodes in the hidden layer which is mostly heuristic in case of other proposed methods. A hybrid optimization technique based on genetic algorithm (GA) and Broyden, Fletcher, Goldfarb, and Shanno (BFGS) Quasi-Newton algorithm is used for faster and effective training of the network. Kernel density estimation is used for finding the confidence limits. The method is applied for detecting the fault in a simulated continuous stirred-tank reactor (CSTR). The result shows that the proposed method is excellent for process monitoring in non-linear systems.

Keywords: Fault Detection, Fault identification, Radial base function networks, Genetic algorithms

1. INTRODUCTION

The advent of faster and more reliable computer systems has revolutionized the manner in which industrial processes are monitored and controlled. Once thought of as just data logging and storage units, these computer systems now perform sophisticated computer-based control strategies, and real-time simulation and optimisation. These advances have resulted in the generation of a large amount of process data, yet the task of interpreting and analysing this data is daunting. The most used method for the data reduction is principal component analysis (PCA). The major drawback of this method is that it assumes the linear correlation between the data which is not always true in case of process data which are generally nonlinearly correlated. Many methodologies have been proposed for nonlinear principal component analysis (NLPCA). Kramer (1991) proposed a NLPCA based on five layer auto associative neural networks. The bottleneck layer which is smaller compared to input and output layer is forced to develop a compact representation of the input data. Since the number of layers is five, it is difficult to train the network. Also, it is difficult to determine the number of nodes in hidden layers and the bottleneck layer which is more troublesome in case of large number of variables. Dong and McAvoy (1996) proposed NLPCA based on principal curves and neural networks. The principal curves method suggested by Hastie and Stuetzle (1989) was used to calculate the associated score and corrected data point for each original data point. But, since principal curve method does not produce a nonlinear principal component in the sense of principal loading, Dong and McAvoy (1996) developed an alternative approach based on multi layer perceptron to model the calculated data.

Most of the principal curve methods (Hastie and Stuetzle, 1989; Kegl, *et al.*, 2000) consist of a combination of 'local models' that are related by a fixed topology. It is suggested that these methods exhibit poor performance when the data are concentrated around a highly curve or self intersection curve (Verbeek, *et al.*, 2002). This is due to the fixed topology among the local models and due to bad initialization. Also, the number of local models is not known a priori and an (educated) guess is required for the number of local models (Verbeek, *et al.*, 2002).

On the other hand, RBF network has significant advantage over multilayer perceptron like, faster

convergence, smaller extrapolation errors, higher reliability and a more developed theoretical analysis. An RBF network performs a non-linear transformation from d-dimensional input space to k-dimensional output space which is the basic requirement of the NLPCA.

In this work, a model is developed based on RBF network and PL. The k-segment algorithm proposed by Verbeek, *et al.* (2002) for finding the principal curve is used for fitting the PL. The whole methodology can be divided into two parts. Firstly, the data is fitted to PL, and associated scores and corrected value is found for each data point. Suitable number of non linear principal component is selected based on explained variance. The number of data groups is found out based on the number of segment required to fit the polygon. In the second part, two RBF networks are developed to model the NLPCA based on the output of the previous. The data is projected to a lower dimension feature space by one network whereas the second network is used for reconstruction to the input space.

The remaining part of the paper is organized as follows. In Section 2, a brief description of the Voronoi Regions (VRs) and PL is given. In Section 3, the proposed algorithm is discussed, together with the presentation of its architecture. In Section 4, two simulations are performed to show the model capability to reduce the data dimensionality and for fault detection. Finally, in Section 5 the paper is concluded with a summary of the features of the proposed algorithm and giving the directions for future work on this topic.

2. VORONOI REGIONS AND POLYGONAL LINE

Let $\boldsymbol{X}_n$ be a dataset of n samples in the $\Re^d$ with samples denoted by $\boldsymbol{x}$. VRs are regions whose vectors are given by

$$V_i = \{ \boldsymbol{x}: d(\boldsymbol{x}, \boldsymbol{y}_i) \leq d(\boldsymbol{x}, \boldsymbol{y}_j) \text{ for all } j \in \mathcal{J} \} \qquad (1)$$

where $d(\boldsymbol{x}, \boldsymbol{y}_i)$ is a distortion measure on the input/output vector space and $\mathcal{J}$ is the index set.
A line s can be defined as:

$$s = \{ s(t) \mid t \in \Re \}, \text{ where } s(t) = c + ut \qquad (2)$$

The distortion measure can be represented as the distance of a point $\boldsymbol{x}$ to a line s, i.e.

$$d(\boldsymbol{x}, s) = \inf_{t \in R} || s(t) - \boldsymbol{x} || \qquad (3)$$

Therefore, the VRs $V_1, V_2, \ldots, V_p$ can be defined as

$$V_i = \{ \boldsymbol{x} \in \boldsymbol{X}_n \mid i = \arg \min_j d(\boldsymbol{x}, s_j) \} \qquad (4)$$

i.e. V_i contains all the datapoints for which the ith line is the closest. The first step in finding the PL is to find k-lines $s_1, s_2, \ldots, s_k$ that minimizes the total squared distance of all points to their closest lines:

$$\sum_{i=1}^{k} \sum_{x \in V_i} d(x, s_i)^2 \qquad (5)$$

To find lines that are local optima of (5), *k*-lines with random orientation and locations are introduced. The VRs are determined and the lines are replaced by the shortest segment of its first principal component (PC) of their VR such that all orthogonal projection to the first PC of the points in the VR are included in the segment.

Sometimes due to the local minima of (5), the poor performance of the algorithm is observed. It is avoided by using the segments of the first PC that are cut off at $3\sigma/2$ from the centroid of the VR, where σ^2 is the variance along the first PC. It is checked whether (5) is decreased or not. If not, the segment that includes all projections to the first PC is used to obtain the guaranteed decrease of (5). These steps are repeated till the convergence.

Since the optimum number of segments is not known in advance to model the data, an incremental strategy is used. The number of segments are increased and optimised till some performance criterion is met.

For determining the position of point where to insert the new segment, the decrease in (5) is computed if a zero length segment is placed on x_i for each x_i in the data set. The new segment is inserted along the first PC of that VR. After the insertion the previous steps are followed for the optimization.

The next step is to link the segments together to form a PL. This goal is achieved through the graph theory concept. Let $G = (V, E)$ is a fully connected graph, where the set of vertices V consists of the $2k$ end-points of the *k*-segments. A set of edges $A \subset E$ which contains all the edges that corresponds to the segments is defined. A sequence of edges $\{(v_0, v_1), (v_1, v_2), \ldots (v_{m-1}, v_m)\}$ in which all edges are distinct is called a 'path'. A path is open if $v_0 \neq v_m$. An open path that passes through every vertex in the graph exactly once is called a 'Hamiltonian path' (HP). The cost of a path P is represented as:

$$l(P) + \lambda a(P), \qquad (6)$$

with $0 \leq \lambda \in \Re$ being a parameter to be set by the user. The term $l(P)$ denotes the length of the edges in P. The length of an edge $e = (v_i, v_j)$ is taken as the Euclidean distance between its vertices:

$$l(e) = || v_i - v_j ||, \qquad (7)$$

The second term *a(P)* is a penalty term equal to sum of the angles between the adjacent edges. The parameter λ controls the trade-off between preferring short paths and paths that do not contain sharp edges. The smaller λ, the smaller is the preference for the smooth curve. The optimum path *P* will be that HP for which the total cost of the path is minimum. An initial PL is constructed using the greedy strategy.

Using the incremental method described above, a sequence of PL with increasing number of segments are obtained. The objective is to find that PL which maximizes the log-likelihood of the data.

Consider a PL of length *l* as a continuous arc length parameterized one dimensional latent variable *t* embedded in $\Re^d$, where the embedding is given by $f: [0, l] \rightarrow \Re^d$. A uniform distribution $p(t) = 1/l$ is assumed on *t* for $t \in [0,l]$. Furthermore, let $p(x|t)$ is a spherical Gaussian distribution with mean point *t* on the PL and covariance matrix $\sigma^2 I$, where *I* is the *dxd* identity matrix. For a latent variable distributed uniformly over a line segment *s* of length $l \geq \sigma$, the negative log-likelihood for a point $\boldsymbol{x}$ can be roughly approximated by

$$\log l + d(s, \boldsymbol{x})^2 / (2\sigma^2) + c, \qquad (8)$$

where *c* is a constant dependent on σ. The effect of higher density occurring at one side of the PL at places where different non-parallel segments are connected is neglected. Hence, the total log-likelihood of the data is approximated as

$$n \log l + \sum_{i=1}^{k} \sum_{x \in V_i} d(s_i, \boldsymbol{x})^2 / (2\sigma^2), \qquad (9)$$

where *l* is the total length of the PL. The different stopping criterion could be used to keep inserting segments until the objective function (9) reaches its first minimum or some limit k_{max} on *k* is reached.

3. THE PROPOSED METHODOLOGY

3.1 Dimensionality Reduction and reconstruction

The data is fitted to a PL using the method discussed in section 2. This PL gives a generalization of the first linear principal component. Every data points are projected orthogonally onto the PL. Thus for every datapoints there are corresponding lengths $t_1, t_2, \ldots, t_n$ along the curve where *n* is the number of data points in *d*-dimensional space (Fig. 1). In analogy to LPCA, this length represents the non linear scores of the data points. Thus the sample vector can be represented as

$$\boldsymbol{X} = f_1(t(\boldsymbol{X})) + \boldsymbol{E_1} \qquad (10)$$

where *t* is the non linear principal component score and $\boldsymbol{E_1}$ is the residual vector.

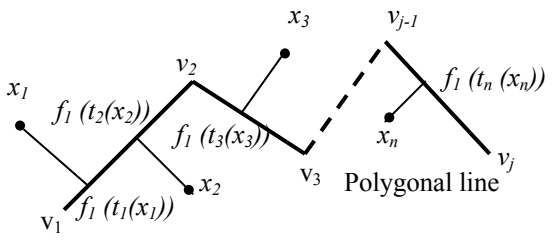


Fig. 1. Projection of the data point on the polygonal line

The next non linear component score can be found by projecting the data points of $\boldsymbol{E_1}$ on the PL constructed using $\boldsymbol{E_1}$. It is found that the first few nonlinear principal components explain most of the variance of the dataset. Though, this method is quite effective in reducing the data dimensionality, it is to be noted that $\boldsymbol{f}$ has no parametric form**,** and it is quite cumbersome and memory expensive to use it for online application. RBF network is used to find out the nonlinear correlation between the original dataset and the reduced space. It is different from the multi layer perceptron in terms of the activation function being used. It has significant advantage over multi layer perceptron like, faster convergence, smaller extrapolation errors, higher reliability and a more developed theoretical analysis. An RBF network performs a non-linear transformation from *d*-dimensional input space to *k*-dimensional output space. There are different paradigms for RBF network that determines the training strategy. In the simplest case, all the training parameters are constant, i.e. no training is required. In the second and more flexible model, only the weights at the output layer are trained. In this case, each node at the hidden layer represents a data vector, which may be too many for large data vectors (Looney, 1996). The third and more flexible approach is to train all the parameters. But still the issue of number of neurons in the hidden layer is based on heuristic or cross validation schemes which become a big problem in case of large process variables.

Since the segments constructing the PL are fitted based on the VRs, each of these segments represents a group of data points. Therefore, it is proposed that each group can be mapped as a neuron in the hidden layer of the network. The centre of the data points represents the centre of the RBF whereas the standard deviation determines the spread. The number of neurons in the output layer is decided by the explained variance of the nonlinear principal components. This gives a well defined architecture of the network based on the theoretical background. Let *k* non linear principal components explain most of the variance of the original dataset of *d* – dimension.

If the dataset is divided into p groups, the architecture of the network will be d-p-k RBF network as shown in fig. 2.

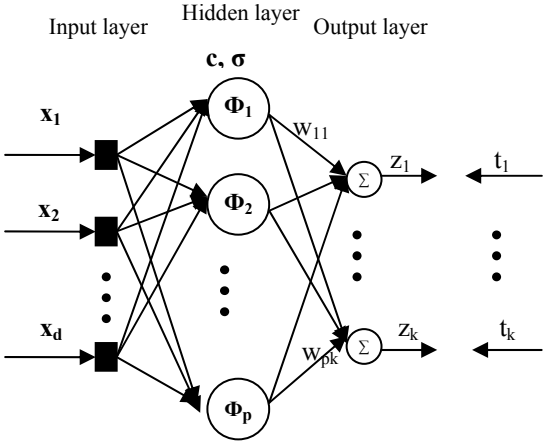


Fig. 2. Architecture of the RBFN for dimensionality reduction (Reduction RBFN).

The Gaussian RBF with equal spread in all the direction is defined as:

$$\varphi(\boldsymbol{x};\boldsymbol{c},\sigma) = exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}\|^2}{2\sigma^2}\right) \tag{11}$$

where $\|\boldsymbol{x}-\boldsymbol{c}\|$ is the Euclidean distance of $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ from the vector center $\boldsymbol{c} = (c_1, c_2 \ldots c_d)$. When the spread of the data points is not uniform, an elliptical Gaussian RBF takes the form

$$\varphi(\boldsymbol{x};\boldsymbol{c},\boldsymbol{\sigma}) = exp\left(-\left[\frac{(x_1-c_1)^2}{2\sigma_1^2}+\ldots+\frac{(x_d-c_d)^2}{2\sigma_d^2}\right]\right) \tag{12}$$

To ensure the optimum receptive field of each of the neuron in the hidden layer, a heuristic approach of using a spread of $3\sigma/2$ is considered, where σ is the standard deviation of the dataset in the group. If $\boldsymbol{W} = (w_{i1}, w_{i2}, \ldots, w_{ik})$ for $i = 1, 2, \ldots, p$ is the weight matrix of the output layer, the output of the network is defined as

$$z_j = f(\boldsymbol{c},\sigma,\boldsymbol{w}) = \left[\sum_{i=1}^{p} w_{ij}\varphi_i\right] \Big/ \left[\sum_{i=1}^{p} \varphi_i\right],\ j = 1, 2, \ldots, k \tag{13}$$

Hence, the objective function to be minimized for the training of the network is

$$\min_{\boldsymbol{c},\sigma,W} \sum_{s=1}^{n}\sum_{j=1}^{k}(z_j - t_j)_s^{\,2} \tag{14}$$

Since only the weights of the nodes in the output layer are unknown in this case, i.e. $\boldsymbol{c}$ and σ are known, the training of this network is extremely faster compared to the multi layer perceptron. A hybrid training algorithm is used to train the network. In this algorithm, first the global minimum is found using GA. GA can reach the region near an optimum point relatively quickly, but it can take many function evaluations to achieve convergence. Therefore, GA is run for a small number of generations to get near an optimum point. The solution of GA is used by BFGS Quasi-Newton algorithm for more efficient local search.

Another network is required to reconstruct the data from the reduced dimension to the original dimension of the sample space (Reconstruction RBF network). In this work also, the mirror image of the first network is taken as suggested by many researchers (Kramer, 1991; Dong and McAvoy, 1996). These two networks combined together is called self mapping neural network.

3.2 Fault detection and Identification

The data of each group is mapped to the lower dimensional space. The centre and standard deviation of each group is found to get the hidden layer parameters. The output layer parameter i.e. the weight matrix is trained using the hybrid training algorithm for normal dataset. The output of the first network is nonlinear principal component scores on the reduced dimension and the output of the second network is the corrected dataset. The network is trained for the normal dataset. The parameter which is observed for fault detection is squared predicted error (SPE). It is defined as

$$\text{SPE} = \sum_{s=1}^{n}\sum_{j=1}^{k}(z_j - t_j)_s^{\,2} \tag{15}$$

In fault detection, the departure of the process from its normal behaviour needs to be detected. In most of the cases, these decisions are based on the confidence limits. These confidence limits are mainly defined based on two types of approach: parametric approaches and non parametric approaches. In the first case, it is assumed that the data belongs to a known distribution, which is not adequate to approximate the non linear processes. In the second approach, the density function is estimated using an unstructured approach. In this work, the non parametric approach is adopted based on kernel density estimation (Martin and Morris, 1996).
Kernel estimator with kernel K is defined by:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh}\sum_{i=1}^{n}\boldsymbol{K}\left(\frac{\boldsymbol{x}-\boldsymbol{X}_i}{h}\right) \tag{16}$$

where h is the window width, also called the smoothing parameter or bandwidth. The quality of a density estimate is mainly determined by the smoothing parameter. The shape of the bumps is determined by the kernel K but it has minor effect on the quality of the density estimate. The smoothing parameter is selected by least square cross validation (Bowman, 1984). The normal density is considered for the kernel.

The identification of the faulty sensor is done on the basis of their contribution to the total error of self mapping neural network. The contribution to the total error by sensor j can be expressed as

$$e_j = \sum_{s=1}^{n} (z_j - t_j)_s^2 \qquad (17)$$

The contribution by the sensor j can be defined as

$$Cont\_j = \frac{e_j}{\sum_{j=1}^{k} e_j} \times 100 \qquad (18)$$

4. CASE STUDY

4.1 Non Linear Equations

Consider a system with three variables but only one factor (Dong and McAvoy, 1996):

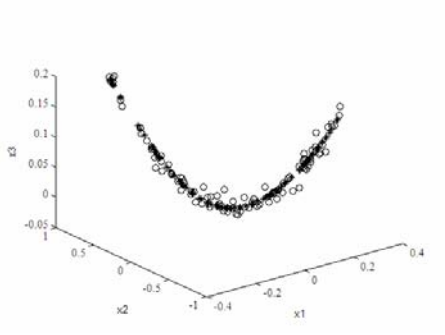
$$\begin{aligned} x_1 &= t_1 + e_1 \\ x_2 &= t^2 - 3t + e_2 \\ x_3 &= -t^3 + 3t^2 + e_3 \end{aligned} \qquad (19)$$

where e_1, e_2, e_3 are independent noise $N(0, 0.01)$, $t \in [0.01, 1]$. 100 samples are generated using these equations. The first nonlinear principal component can explain 99.94% of the variance. The whole dataset is divided into 9 groups. Therefore the structure of the reduction RBF network is 3-9-1, whereas for the reconstruction RBF network, it is 1-9-3. Figure 3(a) shows the comparison between the actual and the predicted data points. It can be seen that the predicted data has an excellent match with the actual datapoints. For checking the detection capability of the methodology, another set of 100 samples are generated by making small changes in x_3. This system can be assumed as the faulty condition in the system.

$$\begin{aligned} x_1 &= t_1 + e_1 \\ x_2 &= t^2 - 3t + e_2 \\ x_3 &= -1.1\,t^3 + 3.2\,t^2 + e_3 \end{aligned} \qquad (20)$$

Fig. 3 (b) shows the SPE of the model. It can be seen that as soon as the fault data is introduced (at observation 100), SPE increases drastically. It shows that the model is capable of detecting even the small fault into the system.

(a)



(b)

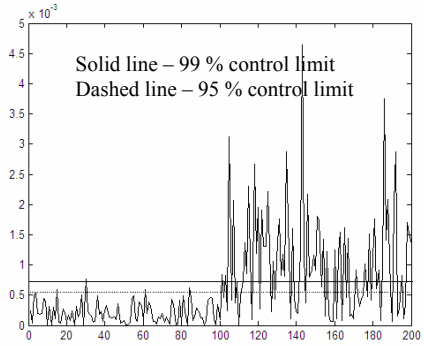


Fig. 3. (a) Comparison between the actual (o) and the predicted (*) values. (b) Square Predicted Error (SPE) using the model.

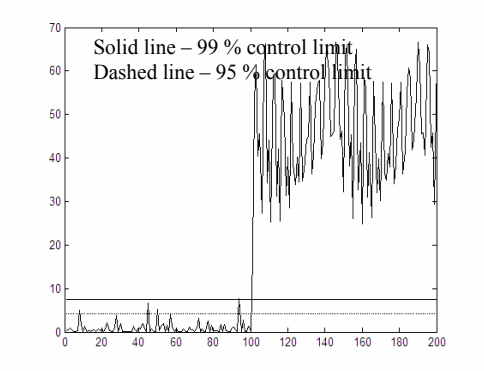
4.2 CSTR Case Study

To show the model capability for detecting fault in industrial system, a simple CSTR system ((Luyben, 1973) is considered. An irreversible exothermic reaction converting reactant A into product B takes place inside the reactor. Nine variables (inputs: Fin (volume flow rate), Tin (temperature), Cain (concentration of component A), Fcin (volume flow rate of incoming cooling stream) and F (volume flow rate of output stream); outputs: F (volume flow rate), T (temperature), Ca (concentration of A), V (volume of content of reaction vessel), and Tc,out (temperature of outgoing cooling stream) are considered for this study. The noise *S(0, 5% of actual value)* is introduced in each input to make it more realistic. The simulink of MATLAB is used to develop the CSTR model. 100 data points at 5 seconds interval for the normal condition is generated. Two faults are introduced into the system. In the first case, the value of Fin is increased by 20% of its actual value (process upset) whereas in the second case, a random number of amplitude 20% of actual value of Fin is introduced (sensor failure). 100 data points at 5 seconds interval for each of these cases are generated.

The explained variance by the first non linear principal component is 98.83% whereas four linear

principal components can only explain 96.34% variance. Fig. 4(a) and 4(b) shows the SPE of the normal and faulty conditions for both of these cases respectively. It can be seen that in both the cases, the proposed model is able to detect the fault the time it is introduced. Since in both the cases the SPE in the faulty condition is much more higher compared to the normal condition it is easy to detect the fault.

(a)



(b)

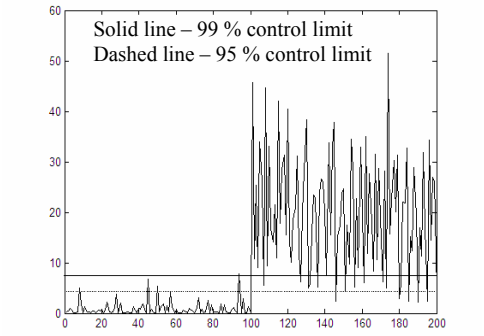


Fig. 4. The SPE using one non linear principal component (a) Process upset (b) Sensor failure.

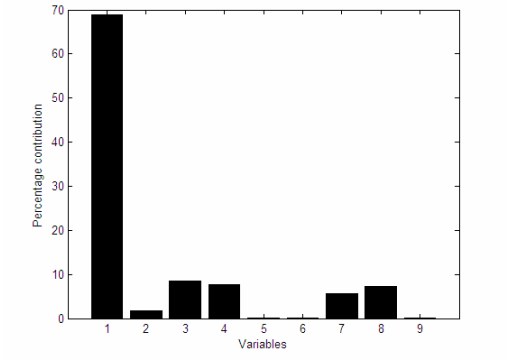


Fig. 5. Contribution plot at the instance when fault was introduced in the system.

Fig. 5 shows the contribution plot at the instant, when the fault was introduced into the system and it clearly indicates that sensor 1 i.e. the volume flow rate meter is the faulty sensor.

An integrated approach of this methodology with the pattern recognition techniques like self organizing map, learning vector quantization etc. can be used to isolate the process upset from the sensor failure. This will be a part of future work.

5. CONCLUSION AND FUTURE WORK

In this work, a network is proposed to model the large dataset into a lower dimension. Also, a scheme to decide the number of nodes in hidden layer and output layer is explained. Since, RBF network is a robust network for function approximation, it is vastly used for feature extraction, pattern recognition and time series predictions. Therefore, the proposed methodology can be used for developing a robust architecture of the network which will be more efficient and theoretically explained. Also, a hybrid scheme for faster and accurate training of this type of network is used. The application of the proposed methodology is shown for fault detection and identification. This work will be the part of an integrated framework for total process control and supervision.

6. REFERENCES

Bowman, A. W. (1984). Alternative method of cross-validation for the smoothing of density estimate, *Biometrika*, **71**, 353-360.

Dong, D. and T.J. McAvoy (1996). Nonlinear Principal Component Analysis-Based on Principal Curves and Neural Networks. *Computers Chem.. Engng.*, **20**, 65-78.

Hastie, T. and W. Stuetzle (1989). Principal Curves. *J. Am. Stat. Ass.*, **84**, 502-516.

Kegl, B., A. Krzyzak, T. Linda and K. Zeger (2000). Learning and Design of Principal Curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **22**,281-297.

Kramer, M. A. (1991). Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. *AIChE Journal*, **37**, 233-243.

Looney, C.G. (1996). *Pattern Recognition Using Neural Networks*. Oxford University Press, Oxford.

Luyben, W. L. (1973). *Process Modeling, Simulation, and Control for Chemical Engineers.* McGraw-Hill Book Company, USA.

Martin, E. B. and Morris A. J. (1996). Non - parametric confidence bounds for process performance monitoring charts. *J. Proc. Cont.*, **6**, 349-358.

Verbeek, J. J., N. Vlassis and B. Krose (2002), A k-segments algorithm for finding principal curves. *Pattern Recognitions Letters*, **23**, 1009-1017.