DATA DRIVEN SOFT SENSOR DESIGN: JUST-IN-TIME AND ADAPTIVE MODELS

by

Shekhar Sharma

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Process Control

Department of Chemical and Materials Engineering

University of Alberta

Abstract

A number of industrial processes involve variables that cannot be reliably measured in real time using online sensors. Many such variables are required as inputs in control schemes to ensure safe and efficient plant operation. Laboratory analysis, which is a reliable method of measuring these variables, is slow and infrequent. Thus, mathematical models called soft sensors which can estimate these hard to measure variables from the abundantly available online process measurements have been used in a number of industrial applications. Among the various soft sensor applications of online prediction, process monitoring, fault detection and isolation, the focus of this thesis is on online prediction and parameter estimation applications.

Just-In-Time (JIT) modeling is a unique framework wherein a local model is created every time a prediction is required. One of the most critical components of JIT models is the similarity criterion which determines the data used in the local models and their associated weights. To handle nonlinear and time varying systems simultaneously under the JIT framework, a new similarity metric which incorporates time, along with the traditional space distance, to evaluate sample weights, is proposed. Further, a query based method to determine the bandwidth of the local models adaptively, as an alternative to the offline global method, is also developed.

Next, the distance-angle similarity criterion used in modeling dynamic systems under the JIT technique is studied. An improved weighing scheme is then proposed which enables a more accurate selection of data for local modeling and provides a better interpretation of results. Again, for this proposed weighing scheme also, an alternative to the global bandwidth estimation, called the point-based method, is proposed.

In the field of online soft sensor prediction and parameter estimation applications, adaptive linear regression algorithms such as recursive least squares and moving win-

dow least squares are widely used because of their simplicity and ease of implementation. However, these methods are not robust to outlying values. We develop a new robust and adaptive algorithm with a cautious parameter update strategy. The proposed algorithm is also quite flexible and a number of variants are easily formulated.

Finally, advantages of the methods are clearly illustrated by applications to numerical examples, experimental data and industrial case studies.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to Dr. Biao Huang, without whose supervision and guidance this thesis would not have been possible. For his encouraging words as well as constructive criticism over the past two years, which helped me to complete this work, I am truly grateful. Besides work related issues, his kind and understanding attitude towards personal matters is also greatly appreciated.

The process control group deserves special mention here. Besides providing a great learning atmosphere, the critical and insightful feedback received during the various group meetings and presentations helped develop and shape this work. I would like to thank Swanand Khare for his advice and inputs in tackling the obstacles encountered during the completion of this thesis. I would also like to acknowledge the help given by Alireza Fatehi, Rahul Raveendran, Rishik Ranjan and Yaojie Lu.

I extend my gratitude to the Department of Chemical and Materials Engineering, University of Alberta, for giving me the chance to pursue this Masters degree. It has been a life enriching experience which I will cherish forever. Financial support from Natural Sciences and Engineering Research Council (NSERC) of Canada and Alberta Innovates Technology Futures (AITF) is also greatly acknowledged.

Finally, I would like to thank my friends and family for their continued love and support throughout my graduate program.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In recent years, the development and application of soft sensors has gained increasing attention. Modern industrial processes are data rich and a large number of fast rate online process measurements are recorded and stored for the purpose of carrying out detailed analysis. However, there are many critical variables that cannot be measured reliably. The existing online analyzers generally have high maintenance requirements and are not reliable. An alternative to online analyzers is laboratory analysis, but the low frequency of sampling and testing presents challenges. Besides, typically in industries, such procedures are a process hazard because of high pressure/temperature conditions, or properties of the process fluids which are generally flammable or corrosive in nature. Hence, availability of reliable and fast rate measurements of these hard to measure variables without depending on laboratory analysis would not only increase production efficiency but also reduce the risks associated with such sampling procedures. Soft sensors have, over the years, proven to be a suitable solution. Essentially, soft sensors are mathematical models that take the easy to measure variables as input and predict the hard to measure variables. Based on collected historical data, once a soft sensor has been validated and approved for online implementation, the laboratory analysis can be done away with or the online analyzer taken offline.

Soft sensors have wide ranging applications from online prediction, process monitoring and process fault detection to sensor fault detection and reconstruction. Consequently, a large array of soft sensors dealing with complex processes have been developed to handle issues such as nonlinearity or multiple operating modes. How-

ever, one of the most important issues surrounding the use of soft sensors is the ability to adapt to changing process conditions. Hence, in order to maintain accuracy, soft sensors either need to have an adaptation mechanism or need to be re-trained periodically when performance objectives are not met.

It is with this perspective that the major portion of the work in this thesis has been carried out, i.e., the development of adaptive models for building soft sensors for online prediction and/or parameter estimation applications.

1.2 Thesis outline and contributions

The thesis is organized as follows.

Chapter 2 provides a brief overview of the Just-In-Time modeling framework. A new similarity metric which takes into account time, along with the traditional Euclidean distance, to calculate weights of the database samples is proposed. This new space-time metric for similarity calculation enables soft sensors based on JIT models to deal with the issues of nonlinearity and time varying property simultaneously. The smoothing parameter in JIT models determines the bandwidth or region of validity of the local model. A query based method to determine this parameter adaptively is further developed. Applications, with both ordinary least squares and partial least squares as the local models, are provided to evaluate the methods.

Continuing with the theme of JIT modeling, the distance-angle similarity criterion used to calculate sample weights is introduced in Chapter 3. The shortcomings of this criterion are highlighted and a more general weight formulation which provides a better representation of the process is proposed. Further, point-based estimation as an alternative to the global method of smoothing/bandwidth parameter selection is explored. The chapter ends with the applications section where the different methods are compared.

In Chapter 4, a novel adaptive linear regression algorithm for online soft sensor prediction and parameter estimation applications is proposed. The new algorithm is robust to outliers in the output, or predicted variable, and follows a cautious parameter update strategy so that minor process disturbances do not affect prediction accuracy. The methods are then applied to numerically simulated and industrial case

studies and a detailed analysis of the results carried out.

Finally, Chapter 5 summarizes the thesis and highlights some areas for future work and improvement. In all chapters, the performance of the different methods is compared based on the correlation coefficient, R, and the root mean square error of prediction, rmsep.

Chapter 2

Space-Time Similarity Criterion for Just-In-Time Modeling of Time Varying Systems*

2.1 Introduction

In this chapter, we discuss the Just-In-Time or JIT modeling framework and propose a novel way to improve its ability to handle time varying systems. One of the key features of JIT modeling methods [1], also known as lazy learning, memory based learning or locally weighted learning [2, 3], is that model building is delayed until the query variable for which prediction is required, is received. This is in sharp contrast to the offline modeling approach, where a single or multiple models are built on available historical data. These offline models extract all the meaningful information from the historical data which can then be discarded. We carry this discussion forward with the focus on application of these modeling methods for building soft sensors or inferential sensors [4]. In recent years, there has been an increasing interest in the development of soft sensors to provide online estimates of otherwise hard to measure quality variables [5, 6]. The measurement of these variables can then be used for a variety of applications from online prediction, process monitoring and process fault detection to sensor fault detection and reconstruction [7].

Linear multivariate modeling techniques such as principal component analysis (PCA) and partial least squares (PLS) have been widely used to build soft sensors for

*This chapter is an extended version of the under review paper: Shekhar Sharma, Swanand Khare, and Biao Huang. Just-In-Time modeling with space-time metric for similarity calculation. *AIChE Journal* (submitted, 2015).

monitoring and control applications [8]. However, since many systems encountered in the process industry are nonlinear, methods such as neural-networks [9], support vector regression (SVR) [10], polynomial functions [11] and fuzzy set [12] have also been used for soft sensor applications. These models are what is called global models and have certain drawbacks. Finding a suitable model structure and selecting the optimal parameters of the model is complex, especially if the process consists of multiple modes. An alternative to global modeling is the use of relatively simple local models to approximate a nonlinear system at different operating points or regimes. T-S fuzzy model and neural fuzzy network are two such modeling techniques. However, this approach has the drawback that expert prior knowledge is required to partition the operating space [12, 13].

Irrespective of what type of model is used for building the soft sensor , the initial step consists of selecting the historical data and pre-processing it to address issues such as missing data, outlier detection and replacement, and selection of relevant variables [5]. A suitable model is then trained on this data and deployed for online application. Models built using this strategy are called offline models since the model building stops once online implementation has begun. However, offline approach works well only when the historical data is representative of all the possible operating modes and drifts of the process plant [7]. This is almost certainly never the case because in most industries, processes exhibit time varying characteristics due to a variety of reasons such as catalyst activity changes, equipment aging and changes of raw materials [13]. Hence, the performance of offline models deteriorates over a period of time. It has been pointed out in literature that one of the key issues surrounding the use of soft sensors is their ability to cope with these changes in process characteristics [5, 13, 14]. Therefore, a number of adaptive techniques to deal with this time varying issue have been proposed in the literature.

It is clear that soft sensors need to have an adaptation mechanism to be able to maintain accuracy for long periods. These mechanisms have been broadly categorized into three different types [15]:

- Instance selection or moving window techniques
- Instance weighting or recursive techniques

- Ensemble methods

A detailed description of the above approaches and review of other issues dealing with adaptation mechanisms in data driven soft sensors has been provided in [7]. We briefly discuss some points associated with adaptive soft sensors here. Given their simplicity and low computational burden, adaptive versions of linear models using the moving window technique and recursive technique are widely used. The block-wise moving window and the recursive versions of linear least squares, principle component analysis (PCA) and partial least squares (PLS) are among the most popular adaptive soft sensors. Versions of adaptive PCA and PLS, modified to handle nonlinearity, such as the moving window kernel PCA [16] and the recursive nonlinear PLS [17] have also been proposed. On the other hand, adaptation of nonlinear models is generally difficult. For example, nonlinear modeling approaches such as fuzzy set, artificial neural networks (ANN) or neuro fuzzy networks, which are one of the most popular nonlinear types, are not easy to adapt due to the difficulties of model structure selection and computational complexity involved in training [7, 12]. Support vector machines (SVM) are an alternative to ANN and possess better generalization property [13]. Recently, adaptive versions of SVM such as online kernel learning (OKL) algorithm [18] and recursive SVR have also been proposed.

A completely different approach to handle the above mentioned issues of nonlinearity and time varying property is the Just-In-Time (JIT) framework of modeling. The first step for building a JIT based soft sensor is similar to the one mentioned before. Available historical data is collected and pre-processed. Next, model building is delayed until the output for a query variable is requested. Hence, this type of models are also called model-on-demand since no offline model exists and no data processing takes place until a query is received [12]. Generally, the models built within the JIT framework exhibit a local structure. Only a subset of the historical data which is most relevant to the query variable is used for model building. Since typical JIT models employ a local model structure, they are able to handle nonlinear systems and track abrupt process changes as well [13, 14]. A number of modeling methods can be employed within the JIT framework [2]. Among the most popular ones are the locally weighted versions of linear least squares and partial least squares. Specifically, PLS

under the JIT framework has been used for several industrial applications including those related to near infrared spectroscopy [19]. Nonlinear models such as SVR and least squares support vector regression (LSSVR) have also been used under the JIT method [13]. However, since JIT modeling involves performing all data processing online at the time of prediction, it is computationally heavy. A major portion of the lookup cost is the cost associated with the local model training. In this regard, locally linear models with their low computation provide a significant benefit over nonlinear models.

The rest of the chapter proceeds as follows. In Section 2.2, a general overview of JIT modeling is provided which lays the basis for the next section. In Section 2.3, we describe two local JIT methods and discuss the problem of handling nonlinear systems with time varying parameters under the JIT framework. An existing method in literature to handle the issue is described and its shortcomings discussed. A novel method of selecting the relevant data set for local model building to address the highlighted issues is then proposed. Further, two techniques for estimating the bandwidth of the local models are also discussed. Advantages of the proposed methods are demonstrated in Section 2.4 by application to a numerical simulation and an industrial case study followed by the concluding remarks in Section 2.5.

2.2 Just-In-Time modeling

In this section, an overview of the JIT modeling framework is provided. Once a query is received, the key steps involved during prediction can be summarized as [12]:

- Select samples in the database which are relevant to the query based on some similarity criterion
- Build a local model on the relevant samples thus selected &
- Calculate the output based on the local model and query

The local model is typically discarded after the prediction has been made. The critical components that facilitate the above steps, and determine the accuracy of prediction are: the historical database, similarity criterion, weight function, local

modeling technique and the database update strategy. Figure 2.1 shows the links between these steps and components. In the following sections, the JIT modeling steps and components are briefly reviewed.

2.2.1 Historical database and database maintenance

Collection and pre-processing of historical data is the first step while building a JIT model. As is the case with offline models, the quality of the JIT model also depends on this historical data. Hence, the database should, if possible, represent all possible operating modes and regimes of the process under study. In the offline modeling approach, the historical data is discarded once the model has been obtained. On the other hand, in the case of JIT models, it is stored and used every time a prediction is required. Also, since only a subset of the data is used for the local model, computational cost and time can be saved in some cases by constructing the database using techniques like k-d trees [2, 3].

Previously, we discussed the need for adaptive models and some related strategies in Section 2.1. JIT based soft sensors should also be able to meet this need. In the absence of a global or multiple offline models, the ability to adapt depends on the how the initial historical database is maintained. Unless the database is adapted to new conditions, the JIT model cannot be expected to maintain accuracy for long periods. Hence, database maintenance is an important component of JIT modeling. In the traditional approach, all new samples are stored into the database, but this approach has two major drawbacks:

- Over time, the size of the database will become very large. This will increase memory requirements on the one hand, and computation costs for similarity calculations on the other.
- Most real life processes show time varying property to some degree and extent. With this approach, very old data that may no longer be relevant can end up participating in the local model and result in decreasing prediction accuracy.

An alternative is the moving window database approach. Here, the database size is limited to a fixed number which is usually based on memory and computation

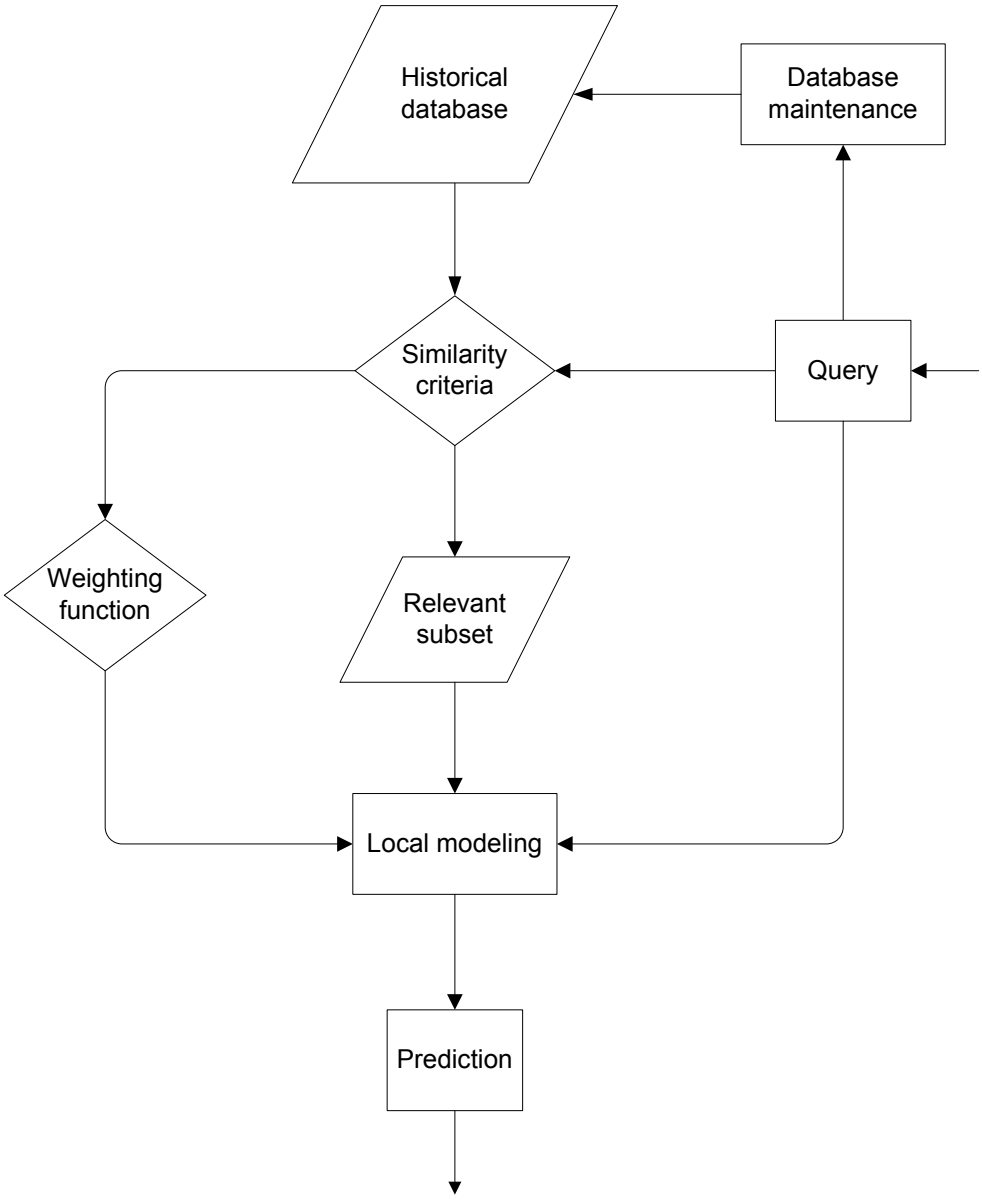

Figure 2.1: JIT flowchart

considerations. Every time a new sample is received, it is stored in the database and the oldest one removed. The size can be chosen to be large enough so that minor process upsets do not replace all useful data and small enough so that memory requirements are not exceeded. This approach is simple and unlike the selective update strategy discussed next, does not increase the computation load during online prediction.

An improved database update strategy is proposed in [20]. In this work, the authors propose a selective update strategy. Only when certain conditions indicating the change of process conditions are met, the database is updated. To prevent database size from increasing continuously, a threshold is specified in advance to keep its size limited.

2.2.2 Similarity criterion and weighting function

Given a historical database and a query, the next step is to select a subset of the data for building the local model. The similarity criterion and weighting function work together to select and prioritize the samples that are relevant or similar to the query variable. The similarity criterion is a qualitative measure of this relevancy or similarity between query and historical samples whereas the weight function is a mathematical function that, given a similarity criterion, assigns weights to the samples. However, both these terms have been used interchangeably in literature.

Similarity Criterion

One of the most commonly used similarity measure is the Euclidean distance (being inversely related to the similarity) [21]:

$$d_{i,E} = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)\,(\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}}} \tag{2.1}$$

where $\mathbf{x}_i$ & $\mathbf{x}_q$ are row vectors representing the i^{th} database sample and the query variable respectively. This measure suffers from the fact that variables with large magnitudes can dominate the distance calculations. Hence, if the variables have different scales of magnitude they are usually mean centered and normalized before distance calculation. A more general form of the Euclidean distance leads to the diagonally weighted Euclidean distance [2, 22]:

$$
\begin{aligned}
d_{i,E_{dw}} &= \sqrt{(\mathbf{x}_i - \mathbf{x}_q)\ \mathrm{D}\ (\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}}} \\
\mathrm{D} &= diag\,(\theta_1, \theta_2, ..\theta_i, ..\theta_m)
\end{aligned}
\tag{2.2}
$$

where θ_i is the weight for distance calculations in the the i$^{\text{th}}$ dimension. Finally, the most general form is given by the fully weighted Euclidean distance [2]:

$$
d_{i,E_w} = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)\ \mathrm{D}\ (\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}}} \tag{2.3}
$$

where D is a positive semi-definite matrix which determines the shape and size of the relevant subset of data. For the sake of clarity, we will refer to the above forms of similarity measures as distance in space. The angle between samples in a dynamic system has also been used as a measure of similarity [12]. The similarity in this case is calculated as:

$$
s_i = \gamma\sqrt{e^{-{d_i}^2}} + (1-\gamma)\cos(\theta_i) \tag{2.4}
$$

where d_i is the Euclidean distance, and θ_i is the the angle between the query, $\mathbf{x}_q$, and database sample, $\mathbf{x}_i$.

$$
\begin{aligned}
\cos(\theta_i) &= \frac{\Delta\mathbf{x}_q {\Delta\mathbf{x}_i}^{\mathrm{T}}}{\parallel \Delta\mathbf{x}_q \parallel_2 . \parallel \Delta\mathbf{x}_i \parallel_2} \\
\Delta\mathbf{x}_q &= \mathbf{x}_q - \mathbf{x}_{q-1},\ \Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1}
\end{aligned}
\tag{2.5}
$$

If $\cos(\theta_i) < 0$, the sample is discarded. γ is a balancing parameter between 0 & 1 which defines the role of distance or angle in the similarity measure. Besides distance in space and angle, the correlation among variables has also been used as a similarity measure in a method called correlation-based JIT modeling (CoJIT) [23].

All the above similarity measures use the input space for similarity calculation. A few methods that utilize output space information for similarity calculation have also been proposed. Wang et al. [24] used the estimated output from an initial global model for the similarity calculation:

$$
s_i = \lambda d_{i,x} + (1-\lambda)\, d_{i,y} \tag{2.6}
$$

$$d_{i,y} = \frac{\mid y_i - \hat{y_q} \mid}{\sum_{i=1}^{N} \mid y_i - \hat{y_q} \mid} \tag{2.7}$$

where $d_{i,x}$ is the Euclidean distance in the input space from $\mathbf{x}_q$ and $\hat{y_q}$ is the estimate corresponding to $\mathbf{x}_q$ calculated from an initial global model.

Weighting Function

Once the similarity criterion has been finalized, the next step is to assign weights to the database samples. Essentially, the weighting function takes the similarity measure of a sample as input and produces the weight of the sample as the output. Atkeson et al. [2] provide a comprehensive review of the weighting functions used in locally weighted learning. Some key properties of the weighting functions are listed below [2]:

- The output of weighting functions and the similarity measure should be directly proportional, i.e., the greater the similarity between samples the higher should be the weight assigned.
- Discontinuities or smoothness of the weighting function are reflected in the discontinuity or smoothness of prediction
- The output of the weighting function should always be non negative

One of the most commonly used weighting function is the Gaussian kernel [2]:

$$e^{-{d_i}^2} \tag{2.8}$$

where d_i is the distance calculated by any one of the previously mentioned measures. A number of variations of the Gaussian kernel have been proposed. Kano et al. [19] have used the following form:

$$e^{-\frac{d_i \phi}{\sigma_{d_i}}} \tag{2.9}$$

where d_i is the Euclidean distance in the input space, σ_{d_i} is the standard deviation of the distances from this query, i.e., $\sigma_{d_i} = \text{std}\ (d_1, d_2, ...d_n)$, n is the total number of

samples and ϕ is the smoothing or bandwidth parameter. ϕ determines the shape of the weighting function and consequently the rate at which the weight falls off with an increase in the distance. For $\phi = 0$, all samples receive an equal weight of 1 and for large ϕ values, the weight falls sharply with increasing distance. For extremely large values of ϕ, all samples other than the nearest one receive a weight tending to zero resulting in nearest neighbor prediction.

There are a large number of weighting functions besides the Gaussian kernel and its variations. Those interested in a more detailed review are referred to [2]. Besides satisfying the properties listed above, the choice of the weighting function does not have a significant impact on the performance of JIT models [2, 3].

2.2.3 Local modeling technique

In this section we discuss how the prioritized samples can be used to make a prediction. One of the simplest technique is the weighted average prediction [2]:

$$\hat{y_q} = \frac{\sum y_i w_i}{\sum w_i} \tag{2.10}$$

where y_i and w_i are the i$^{\text{th}}$ sample output and its weight respectively. Atkeson et al. [2] give a very generalized formulation of the training criterion for the estimation of the local model. Since, for many systems, no single global model is a good fit for the complete data, the local modeling approach is a suitable alternative. This is done by emphasizing data points around the query which results in different models for different queries.

$$C(q) = \sum_i \left[L_q \left(f_q \left(\mathbf{x}_i, \boldsymbol{\beta}_q \right), y_i \right) w_{i,q} \right] \tag{2.11}$$

where $C(q)$ is the training criterion for query $\mathbf{x}_q$, L is the cost function, $f\left(\mathbf{x}_i, \boldsymbol{\beta}_q\right)$ is the prediction function, $\boldsymbol{\beta}_q$ is the local model parameter vector and $w_{i,q}$ is the weight of the i$^{\text{th}}$ sample corresponding to the query, $\mathbf{x}_q$.

A host of cost and prediction functions can be combined to form the training criterion through the above formulation. The training, cost and prediction functions can even be changed from query to query. But for most practical purposes these

are chosen in advance and kept fixed during online operation. Only the local model parameters, $\boldsymbol{\beta}_q$, change with the query. Because of the weighted formulation, the linear models can be emphasized to predict more accurately around the query by giving the accuracy of prediction near it more importance. The nonlinearity exhibited by most industrial processes can be reasonably approximated by linear models around different operating points and since JIT models delay all data processing till prediction time, the simplicity and low computation required for linear models makes them ideal candidates for the JIT framework.

2.3 Just-In-Time modeling with space and time weights

Before proceeding further, we first describe two local modeling techniques that are commonly used under the JIT framework, locally weighted least squares (LWLS) and locally weighted partial least squares (LWPLS). Both of them will be used later on in the applications section.

2.3.1 LWLS and LWPLS

LWLS

Let us assume that the database, query and the weighting matrix are given as: $\mathbf{X} \in R^{n*m}$, $\mathbf{y} \in R^{n*1}$, $\mathbf{x}_q \in R^{1*m}$ & $\mathbf{W} = diag\,(w_1, ..w_i, ..w_n)$, where the input matrix, $\mathbf{X}$, consists of n row vectors representing the observations and the output vector, $\mathbf{y}$, is a column vector of n scalar values and w_i is the weight of the i$^{\text{th}}$ database sample. The equation involved in LWLS to calculate the local regression vector is [3]:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{y} \tag{2.12}$$

LWPLS

In cases where the input dimension is very large or the variables are highly correlated, LWLS becomes unsuitable because of high computation and numerical instability caused by the matrix inversion in Eqn. (2.12). PLS is a technique that is able to handle the above issues. A locally weighted version of PLS called LWPLS suitable

for use under the JIT framework has been derived by Schaal et al [3]. With the query, database and weights defined as in LWLS earlier, the algorithm for making a prediction, $\hat{y}_q$, with the LWPLS method is described below [3]:

1. Mean center the historical data around the query, $\mathbf{x}_q$

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n} w_{ii}\mathbf{x}_i}{\sum_{i=1}^{n} w_{ii}} \tag{2.13}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} w_{ii}\mathbf{y}_i}{\sum_{i=1}^{n} w_{ii}} \tag{2.14}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \mathbf{x}_2 - \bar{\mathbf{x}} \\ \cdots \\ \cdots \\ \cdots \\ \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 - \bar{y} \\ y_2 - \bar{y} \\ \cdots \\ \cdots \\ \cdots \\ y_n - \bar{y} \end{bmatrix} \tag{2.15}$$

2. Calculate the weights (pls weight vector), scores, loadings and the regression parameters of the local PLS model and make the prediction iteratively.

   Repeat the steps below till the specified latent factors, say l, have been extracted:

   Initialize prediction: $\hat{y_q} = \bar{y}$

   For $k = 1 : l$

   Extraction Steps:

$$\mathbf{u}_k = \mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{y} \tag{2.16}$$

$$\mathbf{t}_k = \mathbf{X}\mathbf{u}_k \tag{2.17}$$

$$\mathbf{p}_k = \frac{{\mathbf{t}_k}^{\mathrm{T}}\mathbf{W}\mathbf{X}}{{\mathbf{t}_k}^{\mathrm{T}}\mathbf{W}\mathbf{t}_k} \tag{2.18}$$

$$q_k = \frac{\mathbf{t}_k{}^{\mathrm{T}}\mathbf{W}\mathbf{y}}{\mathbf{t}_k{}^{\mathrm{T}}\mathbf{W}\mathbf{t}_k} \tag{2.19}$$

Deflation Steps:

$$\mathbf{X}_k = \mathbf{X}_k - \mathbf{t}_k\mathbf{p}_k \tag{2.20}$$

$$\mathbf{y}_k = \mathbf{y}_k - \mathbf{t}_k q_k \tag{2.21}$$

Prediction Steps:

$$t_{q,k} = \mathbf{x}_q\mathbf{u}_k \tag{2.22}$$

$$\hat{y}_q = \hat{y}_q + t_{q,k}q_k \tag{2.23}$$

$$\mathbf{x}_q = \mathbf{x}_q - t_{q,k}\mathbf{p}_k \tag{2.24}$$

2.3.2 Space and time weights

Soft sensors need to be adaptive in order to address changes in process characteristics. Various recursive methods which update models by prioritizing new samples have been proposed. However, the most common ones such as recursive least squares, recursive PCA and recursive PLS are all linear. Another issue with recursive methods is that their performance deteriorates if there are sudden changes, such as equipment cleaning, in the process characteristics [14]. This is because there is a significant difference in the query and the most recent database samples, and since the prioritization of recursive methods is based on time only, the data samples in the past that might contain information relevant to the query will always get lower weights whereas recent samples will always get higher weights. On the other hand, because of the use of localized models, JIT based soft sensors are able to handle nonlinearity well. However, there are some drawbacks. Typically, the database size is dictated by

memory and computation requirements associated with distance calculations required for local modeling. As such, if the system under study is time varying, older samples in the database may be relevant in space, but may not be so in time. Therefore, JIT applications generally involve time-invariant static or dynamic nonlinear systems and time varying property is rarely addressed.

RLWPLS

To handle the issue of time varying property and nonlinearity together, a modified JIT technique, called RLWPLS has been proposed by Chen et al. [25]. Consider a moving window database with size n. The weighing scheme in RLWPLS involves dividing the database into two sections, the first containing the recent k samples and the second containing the remaining $n-k$ samples. Next, time weights are assigned to the first section and space weights to the second section. The final weighing matrix for the complete database is obtained by combining the time and space weighted sections using a balancing parameter ρ as follows:

$$\begin{bmatrix} w_s(\mathbf{x}_1) & & & & & \\ & \ddots & & & 0 & \\ & & w_s(\mathbf{x}_{n\text{-}k}) & & & \\ & & & \rho\lambda^{k-1} & & \\ & 0 & & & \ddots & \\ & & & & & \rho\lambda^0 \end{bmatrix}$$

where, w_s is the space weight, normalized between 0 and 1, based on the Euclidean distance of the sample, $\mathbf{x_i}$, from the query, $\mathbf{x_q}$. ρ lies in $[0, 1]$ and denotes the importance of the time weighted section of the database. Finally, λ is the forgetting factor for the time weights calculation. We make a few observations regarding the above approach:

1. The weighing scheme treats nonlinearity and time varying property separately, but not simultaneously. If the system is time varying, older samples should not be given large weights. However, the above algorithm forces time weights of 1

to all the past database samples. On the other hand, there are no space weights for the most recent samples, essentially ignoring any nonlinearity in them.

2. The algorithm uses a fixed bandwidth/smoothing parameter for space weight calculations, thus making it unable to adjust the space weight curves according to the system noise. Even then, 3 parameters are used, all of which are proposed to be selected based on offline optimization which is computationally heavy.

To address the above drawbacks we propose a new generalized framework to incorporate time into the weighing scheme. The essential concept is to give each data sample, space weight as well as time weight, which, working together, prioritize the samples. Considering time as another variable, it can be introduced by simply adding an extra dimension to the input vector. This dimension, the time dimension, can be said to represent the age of the sample. Recent samples have lower age and older samples have higher age. We call the traditional JIT using space weights only as JIT_s and the new formulation using space and time weights as JIT_st to signify the difference.

JIT_st

For a database of size n where the input variable, $\mathbf{x}$, is an m dimensional row vector, the addition of the time dimension is done as:

$$\begin{aligned}
&\mathbf{x}_q = [\mathbf{x}_q, 0] = [x_{q,1}, x_{q,2}, ...x_{q,m}, 0] \\
&\mathbf{x}_n = [\mathbf{x}_n, 0] = [x_{n,1}, x_{n,2}, ...x_{n,m}, 0] \\
&\mathbf{x}_{n-1} = [\mathbf{x}_{n-1}, 1] = [x_{n-1,1}, x_{n-1,2}, ...x_{n-1,m}, 1] \\
&. \\
&\mathbf{x}_i = [\mathbf{x}_i, n-i] = [x_{i,1}, x_{i,2}, ...x_{i,m}, n-i] \\
&. \\
&\mathbf{x}_1 = [\mathbf{x}_1, n-1] = [x_{1,1}, x_{1,2}, ...x_{1,m}, n-1]
\end{aligned} \tag{2.25}$$

The last entry added to the query and the input variable represents the time index. Hence $\mathbf{x}_n$, the latest sample, has the time index of 0 which makes it closest to the

query in time. Similarly, $\mathbf{x}_1$, the oldest sample in the database is farthest in time from $\mathbf{x}_q$. The Euclidean distance of the i$^{\text{th}}$ sample from $\mathbf{x}_q$ is:

$$d^2{}_i = (\mathbf{x}_i - \mathbf{x}_q)(\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}} \tag{2.26}$$

$$d^2{}_i = (x_{i,1} - x_{q,1})^2 + ... (x_{i,m} - x_{q,m})^2 + (n-i)^2 \tag{2.27}$$

The distance without the time dimension is denoted as the distance in space, $d_{i,s}$, and the distance in time as $d_{i,t}$

$$d^2{}_i = d^2{}_{i,s} + d^2{}_{i,t} \tag{2.28}$$

Next, introducing different smoothing/weighting parameters for the space and time dimensions as ϕ_s and ϕ_t we have:

$$d^2{}_i = \phi_s d^2{}_{i,s} + \phi_t d^2{}_{i,t} \tag{2.29}$$

Since the focus here is on time as an added dimension, we have used a common smoothing parameter, ϕ_s, for all the space dimensions in the input variable. Using a different ϕ for every dimension will result in the diagonally weighted Euclidean distance mentioned earlier. Next, using the Gaussian kernel for the weight function we have:

$$w_i = e^{-{d_i}^2} = e^{-\left(\phi_s d^2{}_{i,s} \,+\, \phi_t d^2{}_{i,t}\right)} \tag{2.30}$$

$$w_i = e^{-\phi_s d^2{}_{i,s}} . e^{-\phi_t d^2{}_{i,t}} \tag{2.31}$$

Thus we see that the weight of any sample in the database is determined by two components, distance in space, $d_{i,s}$, and distance in time, $d_{i,t}$. Without loss of generality, one could also write Eqn. (2.31) above as:

$$w_i = e^{-\phi_s d_{i,s}} . e^{-\phi_t d_{i,t}} \tag{2.32}$$

$$w_i = w_{i,s} . w_{i,t} \tag{2.33}$$

We name this new weighting scheme as the space-time metric for similarity calculation. Ultimately, the weight can be expressed as a combination of space weight and time weight, each of which can be calculated using a specific exponential form. The ϕ_s & ϕ_t parameters can be manipulated to control the smoothing along space and time respectively. Table 2.1 shows how the values of the smoothing parameters, ϕ_s and ϕ_t, reflect the degree of nonlinearity and time varying property of the system respectively.

Table 2.1: ϕ_s , ϕ_t interpretation

Case	ϕ_s	ϕ_t	Nonlinearity	Time varying property
1	low	low	low	low
2	low	high	low	high
3	high	low	high	low
4	high	high	high	high

Comparing the approach proposed above with RLWPLS, the advantages become obvious. First, unlike in RLWPLS, every database sample is assigned both, time weight and space weight. Second, whereas RLWPLS introduces 3 additional parameters, k, λ, & ρ, JIT_{st} introduces only 1 additional parameter, ϕ_t (ϕ_s is ignored as it can be added in RLWPLS as well for specifying the bandwidth with respect to the space dimension). This decrease in the number of parameters not only makes offline selection less computationally heavy but also makes adaptive query based parameter selection feasible which is discussed next.

2.3.3 Bandwidth/Smoothing parameter estimation

Given the query variable and the choice of the local model, say LWLS or LWPLS, the value of the bandwidth parameter, ϕ, is required to make a prediction. There are a number of ways to select this value, such as [2]:

1. Fixed bandwidth selection
2. Nearest neighbor bandwidth selection
3. Global bandwidth selection

4. Query-based local bandwidth selection

5. Point-based local bandwidth selection

Among these, we discuss global and query-based bandwidth selection.

Global bandwidth selection

In this method, the value of ϕ is selected by minimizing a cost function, such as the root mean square error of prediction (rmsep), on the training data. It is a popular method and simple to use if the number of smoothing parameters are few. However, the method becomes unwieldy for cases when different smoothing parameters are used for weights on different dimensions.

Query-based bandwidth selection

The methods in this class select the smoothing parameters adaptively for each query. One such technique is based on minimizing the locally weighted leave-one-out cross validation (loocv) error for every query. In the least squares framework, the loocv error [26], and its weighted version [2], for a linear model of the type, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$ are given as:

$$e_l = \frac{1}{\sum_{i=1}^{n} n} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i \boldsymbol{\beta}\,(i)\right)^2 \tag{2.34}$$

$$e_l = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \left(y_i - \mathbf{x}_i \boldsymbol{\beta}_{\boldsymbol{w}}\,(i)\right)^2 \tag{2.35}$$

where e_l denotes the loocv error, n is the sample size and w_i are the weights. $\boldsymbol{\beta}$ and $\boldsymbol{\beta_w}$ are the ordinary and the weighted least squares regression coefficient vectors respectively. The subscript i denotes the i^{th} observation whereas (i) denotes that this observation is not involved in calculations. Hence, $\boldsymbol{\beta}_{\boldsymbol{w}}(i)$, in Eqn. (2.35) is calculated as:

$$\boldsymbol{\beta}_{\boldsymbol{w}}(i) = \left(\mathbf{X}\,(\mathbf{i})^{\mathrm{T}} \mathbf{W}\,(\mathbf{i}) \mathbf{X}\,(\mathbf{i})\right)^{-1} \mathbf{X}\,(\mathbf{i})^{\mathrm{T}} \mathbf{W}\,(\mathbf{i}) \mathbf{y}\,(\mathbf{i}) \tag{2.36}$$

Therefore, to calculate e_l by Eqn. (2.35), one would have to compute the matrix inverse in Eqn. (2.36), $\left(\mathbf{X}\left(\mathbf{i}\right)^{\mathrm{T}}\mathbf{W}\left(\mathbf{i}\right)\mathbf{X}\left(\mathbf{i}\right)\right)^{-1}$, n times for every time an observation is removed. For online applications, this is not practical due to the high computation load. However, using the PRESS statistic [27], one can compute the exact value of e_l without having to perform the calculation given in Eqn. (2.36) n times. The local version of e_l using the PRESS statistic is given as:

$$e_l = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \left(\frac{y_i - \mathbf{x}_i \mathbf{P} \mathbf{X}^{\mathrm{T}} \mathbf{W} \mathbf{y}}{1 - w_i {\mathbf{x}_i}^{\mathrm{T}} \mathbf{P} \mathbf{x}_i} \right)^2 \tag{2.37}$$

where $\mathbf{W}$ is the diagonal weighting matrix with the weights calculated with respect to the given query $\mathbf{x}_q$. $\mathbf{P}\left(= \left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\right)$ is the inverse of the weighted co-variance matrix of all the samples. Hence, in Eqn. (2.37), the matrix inversion has to be performed only once which is a computationally efficient way to compute e_l. We also see that given the query $\mathbf{x}_q$, and the database $(\mathbf{X}, \mathbf{y})$, e_l is dependent only on the smoothing parameter. Hence, for the $\mathrm{JIT}_{\mathrm{st}}$ method proposed earlier, a grid search can be performed over ϕ_s and ϕ_t and the pair that minimizes e_l can be selected as the optimum one. As an example, the time weight curves for a database consisting of 300 samples are shown in Figure 2.2, the distance in time and the corresponding weight being calculated as given in Eqn. (2.28) and Eqn. (2.32) respectively. Figure 2.2 displays 10 weight curves corresponding to 10 ϕ_t values. The ϕ_t range is from 0 representing a time-invariant system to the upper limit of 0.1 at which the time weight of the 250th sample reduces to nearly zero.

For the case of LWPLS however, there is no exact computationally efficient version of e_l like the one above for weighted least squares. In LWPLS, e_l can be approximated under the assumption of independent projections by the following term [3]:

$$e_l \approx \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} \sum_{k=1}^{l} \frac{w_i \; {res^2}_{k,i}}{(1 - w_i \frac{{t^2}_{k,i}}{{\boldsymbol{t}_k}^{\mathrm{T}} \mathbf{W} \boldsymbol{t}_k})^2} \tag{2.38}$$

where w_i are the sample weights making up the diagonal weight matrix $\mathbf{W}$, $\boldsymbol{t}_k$ is the k^{th} score vector and $t_{k,i}$ is the i^{th} element of this vector, and ${res^2}_{k,i}$ is the i^{th} element of the residual $\mathbf{y}$ vector, $(\mathbf{y}_k - \mathbf{t}_k q_k)$. Again, only one LWPLS model has to be computed to calculate e_l corresponding to one ϕ_s , ϕ_t pair. Similar to weighted least squares,

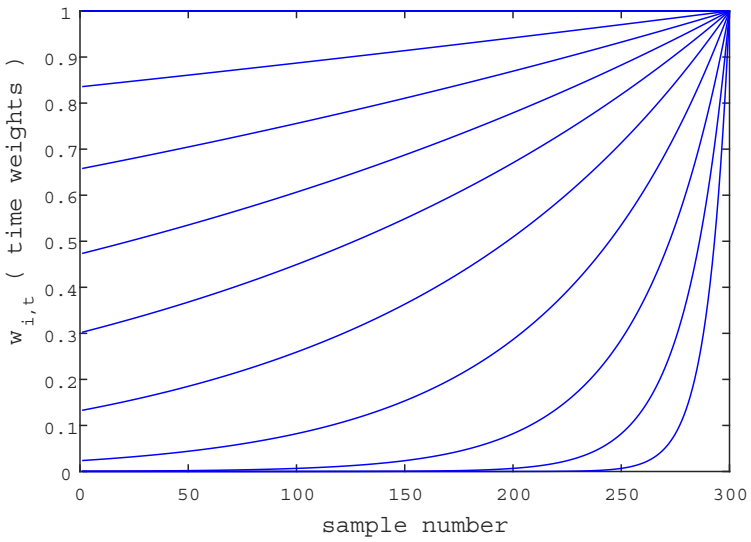

Figure 2.2: Time weight curves variation with ϕ_t. The steepness of the curves increases with increase in ϕ_t. 300:most recent & 1:oldest database sample

the ϕ_s , ϕ_t pair corresponding to the minimum e_l is selected. The greater the number of ϕ_s & ϕ_t pairs on the grid, the higher the computation. The computation load is not a major concern for the global selection method since it is performed offline. For the online case, however, it can be impractical to calculate loocv for every query if the number of pairs is too large. Also, the loocv criterion has some undesirable properties such as the tendency for overfitting [14, 28]. Fortunately, there are certain guidelines that can be used to tackle these issues and expert knowledge can be used to limit the grid search within reasonable bounds. This will be discussed further in the results section.

2.4 Results and discussion

In this section, the proposed JIT method with space and time weights is applied on a simulated and an industrial data set. The results clearly bring out its advantage.

2.4.1 Numerical simulation

Here, a time varying non linear static system is generated as follows:

$$y_i = 3 + a_t \sin x_i + \epsilon_i \tag{2.39}$$

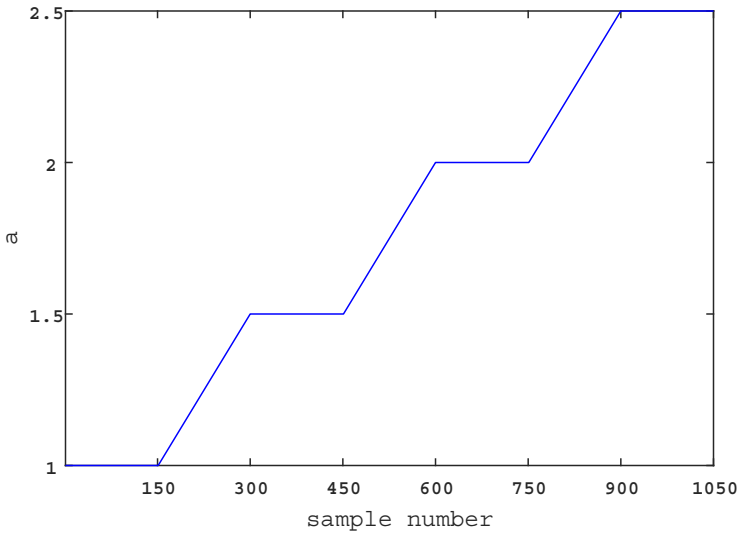

Figure 2.3: a variation

where, x_i, the input, is a uniformly distributed random number in the interval $\left[\frac{\pi}{2} - 0.5, \frac{\pi}{2} + 0.5\right]$, ϵ_i is zero mean white noise, & a_t is a time varying parameter.

A total of 1050 samples are generated for the training and test sets. For prediction during training and testing, the initial 300 points of each set are used as the historical database. Using the moving window approach, the size of the database is restricted to 300, with the newest sample being added and the oldest removed from the database. The value of a_t is varied as shown in Figure 2.3 and Figure 2.4 displays the time varying nonlinear relationship between the noise free training output y, and the training input x. In this numerical example, the relationship between y and x changes from the bottom of Figure 2.4 to the top indicating an increase in nonlinearity with time. Next, LWLS regression is used under the JIT framework for the evaluation of the different methods. The following weighting function is used for the calculation of the space and time weights:

$$w_i = e^{-\phi d_i} \tag{2.40}$$

For JIT_{st}, the sample weight is the product of the space weight and time weight given by Eqn. (2.32).

The following methods are compared.

- JIT_{s}: Traditional JIT with space weights only. Global ϕ_s is obtained offline by

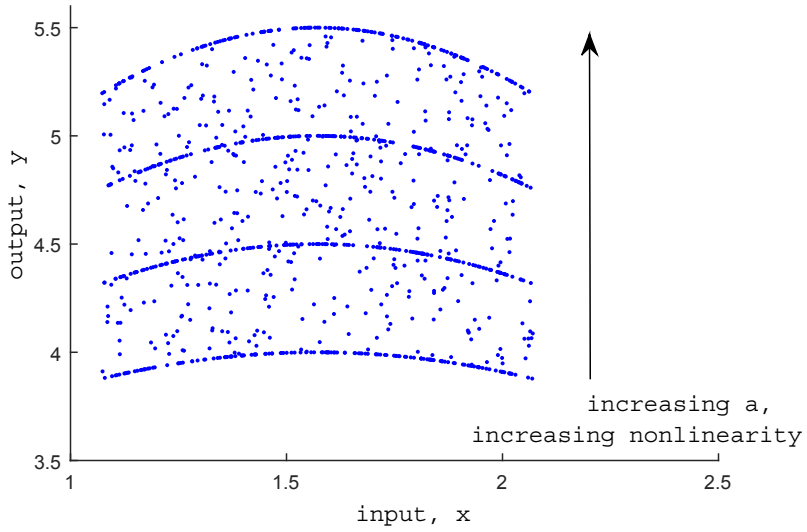

Figure 2.4: Noise free output vs. input

minimizing rmsep on the training set.

- RLWPLS: JIT with space and time weights as proposed by Chen et al. [25], with linear least squares as the local model. k, ρ, λ are obtained offline by minimizing rmsep on the training set.

- JIT$_{\text{st}}$global: JIT with space and time weights. Global ϕ_s & ϕ_t are obtained offline by minimizing rmsep on the training set. A grid search over ϕ_s & ϕ_t pairs is performed for the training set. The pair that minimizes the rmsep is selected as the optimal.

- JIT$_{\text{st}}$press: JIT with space and time weights. ϕ_s & ϕ_t are determined online. For every query, e_l is calculated using the PRESS statistic over a grid of ϕ_s & ϕ_t values. The pair that minimizes this cross validation error is used to predict the response for the given query. The grid used in this numerical simulation is:

  $\phi_s = \{2, 4, ..., 38, 40\}, \qquad \phi_t = \{0, 0.0006, ..., 0.05, 0.1\}$

  The ϕ_t values are the same that represent the time weight curves shown in Figure 2.2.

Table 2.2 presents the test set results. JIT$_{\text{st}}$global handles nonlinearity and time varying property simultaneously and hence performs the best. Let us analyze the

Table 2.2: Test Results: Numerical simulation

Model	ϕ_s	ϕ_t	rmsep
JIT_s	16	-	0.316
RLWPLS ($k = 300, \lambda = 0.97, \rho = 1$)	-	-	0.216
JIT_stglobal	6	0.05	0.204
JIT_stpress	-	-	0.236

weights of JIT_stglobal for one query case where its prediction is much better than JIT_s. The query considered is the 914^{th} and the errors in prediction by JIT_stglobal and JIT_s are 0.22 & 0.62 respectively. Figure 2.5 displays the weights of the 300 samples in the database corresponding to the 914^{th} query under the JIT_stglobal method. The vertical green lines represent space weights and the black line represents the time weight curve corresponding to $\phi_t = 0.05$. Consider the sample highlighted in red, sample 22, in Figure 2.5. The space weight of the sample is nearly one. The y value corresponding to this sample is 4.89 whereas the y value corresponding to the query is 5.53. Hence, considering only space weights can lead to high weights being assigned to samples that should not be included in the modeling due to the time varying nature of the system. The final weights under the JIT_stglobal concept, given by the product of the space and time weights, (all weights are scaled between 0 & 1, the maximum getting 1 and the minimum 0 weight) are shown in Figure 2.6.

We see that the previous database sample with large space weight gets zero weight because of its large age or distance in time. However, it is also noted that it is not necessary that older samples always get lower weight than newer ones. As can be seen in Figure 2.6, some very recent samples do not get high weights because of their large distance in space. Hence, the above formulation is able to handle both, nonlinear and time varying system. Another interesting observation is the result of JIT_stpress. Even without any offline selection of smoothing parameters and a wide grid range of ϕ_s & ϕ_t, it gives a much better result than JIT_s. It shows clearly that the PRESS statistic can be used for adaptive query based selection of both the space and time smoothing parameters. In the case of RLWPLS, training leads to a k value of 300. Since the window size itself is also 300, this indicates that RLWPLS considers only time varying property and ignores the system nonlinearity. This confirms the earlier observation that RLWPLS does not address system nonlinearity and time

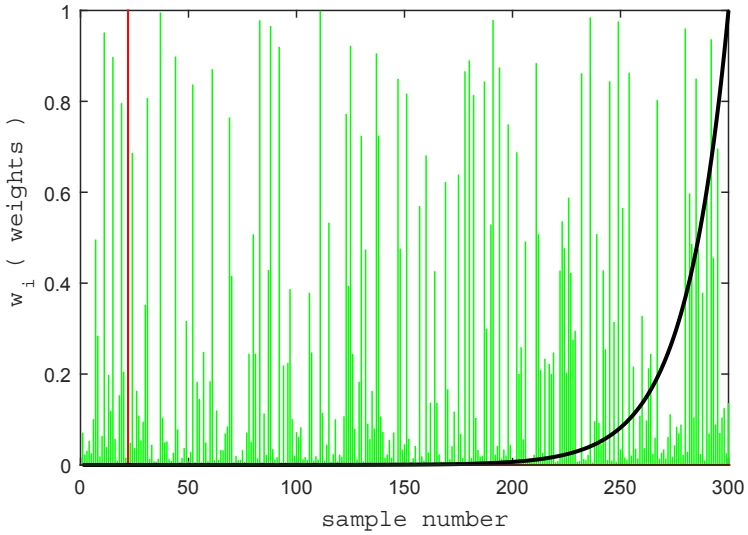

Figure 2.5: Space & Time weights for query 914. Green lines indicate space weights and the black curve represents the time weight curve corresponding to $\phi_t = 0.05$. 300:most recent & 1:oldest database sample

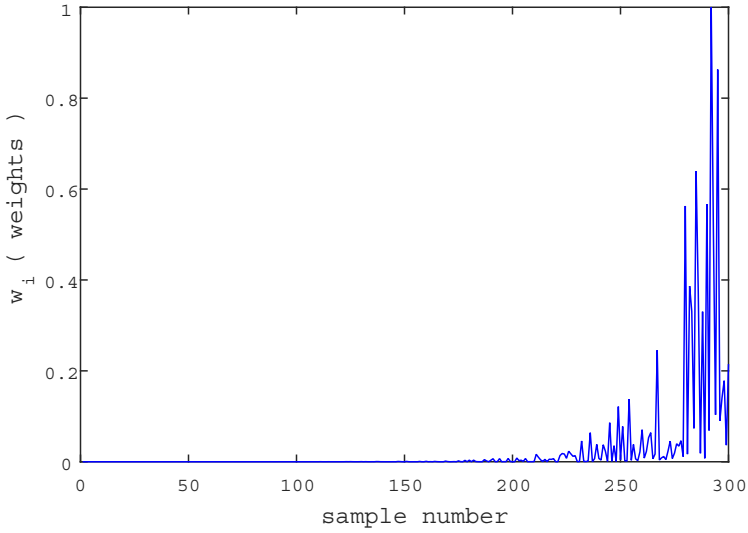

Figure 2.6: Final weights for query 914 under JIT_{st}global

varying property simultaneously but attempts to treat them separately by dividing the database into 2 different sections. Since in this simulation the system is highly time varying, RLWPLS ignores the nonlinearity completely and is consequently less accurate than JIT$_{\text{st}}$global.

2.4.2 Industrial case study

NIR (Near Infrared Spectroscopy) data set for diesel, from a refinery in Edmonton, Canada, is used for performance evaluation of the algorithms in this industrial case study. The data set consists of the absorbance values corresponding to 901 wavelengths from 800 to 1700 nm. The diesel density is the variable of interest. The objective is to be able to predict the density given the absorbance values of a particular sample. Hence, the aborbance values can be treated as the input, $\mathbf{X}$, and the diesel density as the output, $\mathbf{y}$. The spectrum, i.e., the absorbance values, of the sample can be obtained online whereas the corresponding target value, density, is typically measured through offline laboratory analysis. As the frequency of the lab analysis is generally quite low, it is an advantage to be able to predict the density reliably from the available fast rate measurements. Savitzky-Golay method [29] was used to pre-process the data. Outliers were detected and removed based on the 3σ rule [30] and density values were normalized due to proprietary reasons.

Nearly 500 samples form the training and the test sets. As in Section 2.4.1, the initial 300 points are used as the historical database during both training and testing. The moving window database approach with a size of 300 samples is used. The weight function used is as before:

$$w_i = e^{-\phi d_i} \tag{2.41}$$

Because of the large input dimension and highly correlated absorbance values, the LWPLS method with 5 latent factors (based on % variance explained), described in Section 2.3.1, is used. The ϕ_s and ϕ_t grid over which the search for optimal smoothing parameters in JIT$_{\text{st}}$global is carried is:

$$\phi_s = \{25, 50, 75..., 475, 500\}, \qquad \phi_t = \{0, 0.0004, ..., 0.05, 0.1\}$$

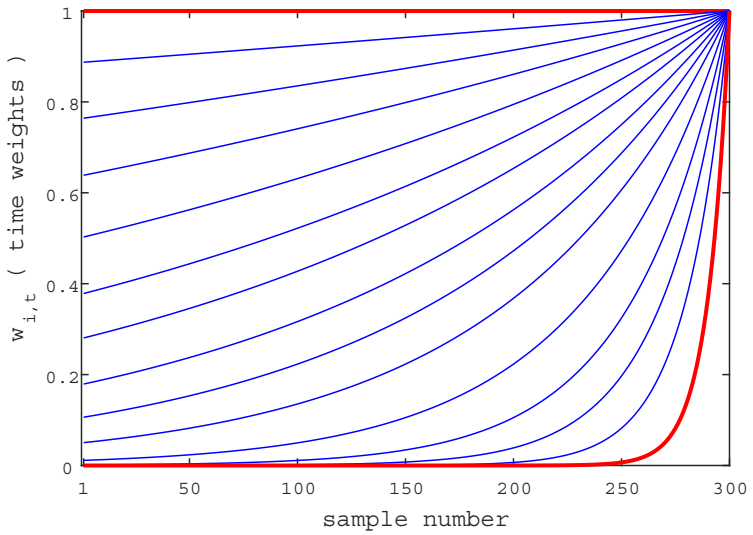

Figure 2.7: Time weight curves variation with ϕ_t. The steepness of the curves increases with increase in ϕ_t. 300:most recent & 1:oldest database sample

Figure 2.7 represents the time weight curves corresponding to the ϕ_t values mentioned above.

With 15 time weight curves and 20 ϕ_s values, there are a total of 300 ϕ_s, ϕ_t pairs. Depending on the available computational capability, this number can be increased/decreased by either increasing/decreasing the ϕ_s, ϕ_t range, or increasing/decreasing the resolution of the grid, i.e., by fitting more/less time weight curves within the extreme two curves (highlighted in red) in Figure 2.7. In this case study, the search with respect to ϕ_t is limited by the value of 0.1, which corresponds to the steepest time weight curve in Figure 2.7. With this curve, the time (and hence the total) weight of the 50^{th} most recent sample reduces to almost zero.

Based on experience and to make the predictions more robust, if it is preferred to include a certain minimum number of samples for modeling, prior knowledge can be used to restrict the grid accordingly. It was pointed earlier that use of loocv for prediction can lead to overfitting. Especially for physical processes, high ϕ values, though reduce the bias in prediction, make the prediction less robust to noise or other disturbances and outliers. Hence, selection of ϕ is a tradeoff between prediction bias and variance, with low ϕ values leading to biased but robust prediction and high

ϕ values leading to a prediction with less bias but higher variance. Consequently, in practical applications, the ϕ value that is selected as the optimum is not usually the one that exactly minimizes the cost function. If an increase in ϕ brings about an insignificant decrease in training error, the lower value of ϕ is selected as the optimum.

Thus, for JIT$_{\text{st}}$press, the optimum ϕ_s, ϕ_t pair is changed only if the loocv decreases by more than 5%. Based on this, for every time curve, starting with the least ϕ_s value, an optimum ϕ_s, ϕ_t pair is selected. 15 such pairs corresponding to the 15 ϕ_t values are so obtained. Next, starting with the ϕ_s, ϕ_t pair corresponding to the least time varying system, i.e. $\phi_t = 0$, as the optimum, it is changed only if a 5 % or greater decrease in loocv is observed along these 15 pairs. The 5 % criterion is general and the results are not very sensitive to this value. One may also use other guidelines for selecting the bandwidth based on loocv while avoiding overfitting. For example, since the variation in the nonlinearity or the time varying property of physical systems is not very large, restricting ϕ_s, ϕ_t values around the global ones obtained from offline optimization can be used as an alternative. Similarly, in the cases of JIT$_{\text{s}}$ & JIT$_{\text{st}}$global, a lower value of ϕ_s and of $\{\phi_s, \phi_t\}$ is selected respectively, if the decrease in rmsep for the training set is not appreciable. Table 2.3 shows the performance of the methods on the test set.

Table 2.3: Test Results: Industrial case study

Model	ϕ_s	ϕ_t	rmsep	R
JIT$_{\text{s}}$	150	-	0.568	0.76
RLWPLS ($k = 275, \lambda = 0.97, \rho = 1$)	-	-	0.554	0.77
JIT$_{\text{st}}$global	150	0.023	0.526	0.79
JIT$_{\text{st}}$press	-	-	0.547	0.78

To analyze the results, a subsection of the predicted density values where JIT$_{\text{st}}$global performs considerably better than the others is displayed in Figure 2.8. It is observed that JIT$_{\text{s}}$ causes a bias in the prediction which is especially strong between the 380^{th} and the 390^{th} samples. Since JIT$_{\text{s}}$ considers only space weights, past data samples which are not relevant, end up getting high weights. Figure 2.9 shows the database weights with respect to 3 such queries, 382, 383 & 384 for JIT$_{\text{s}}$ & JIT$_{\text{st}}$global. Again, based on the results, we see that JIT$_{\text{st}}$global is able to handle time varying property and nonlinearity simultaneously. Old samples are given low time weights as shown by

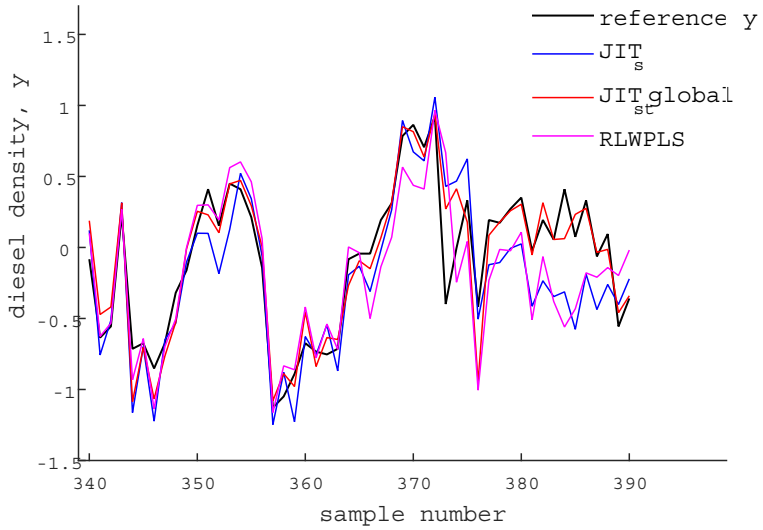

Figure 2.8: Test set prediction performance

the red weight curve in Figure 2.9. However, it is not necessary that older samples will always have lower weights. We can see from Figure 2.9 that some older samples get higher weights than newer samples thus taking into account the system nonlinearity. As a result, JIT_{st}global with time and space weights performs better than JIT_{s} which uses only space weights. JIT_{st}press, which does not have any explicit training period also gives results which are better than JIT_{s}. In fact, JIT_{st}press does not even observe the training set, and still gives reasonable results over a wide range of possible ϕ_s & ϕ_t values.

In the case of RLWPLS, training leads to $k = 275, \lambda = 0.97$ & $\rho = 1$. The sample weights corresponding to the queries 382, 383 and 384 in the case of RLWPLS are shown in Figure 2.10. The weighing scheme demonstrates that RLWPLS treats time varying property and nonlinearity separately. Old samples end up having high weights whereas only a linear model is fit on the most recent samples thus decreasing prediction accuracy. These results clearly demonstrate the superiority of the proposed JIT_{st} method.

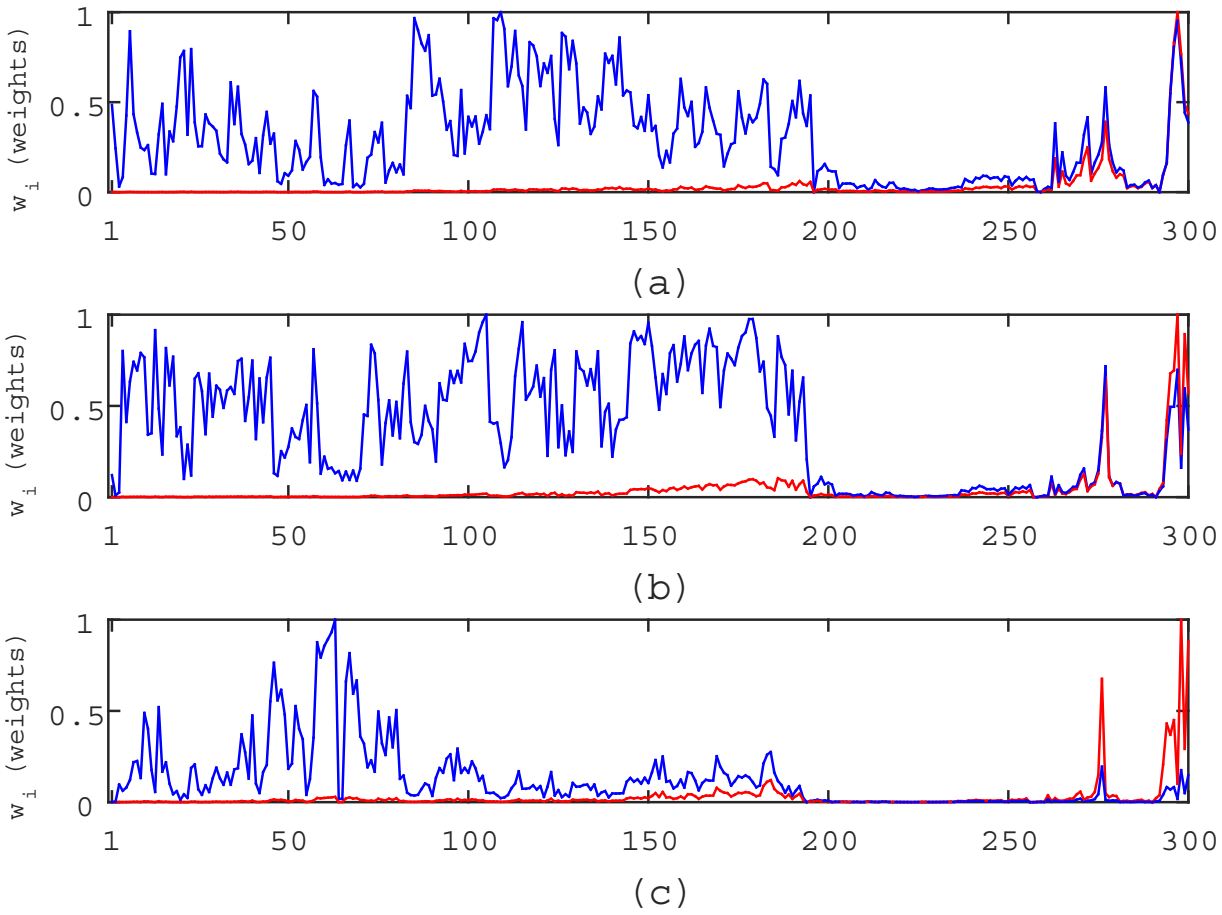

Figure 2.9: Figures (a), (b) & (c) display sample weights for queries 382, 383 & 384 respectively. - JIT_s weights, - JIT_{st}global weights. 300:most recent & 1:oldest database sample.

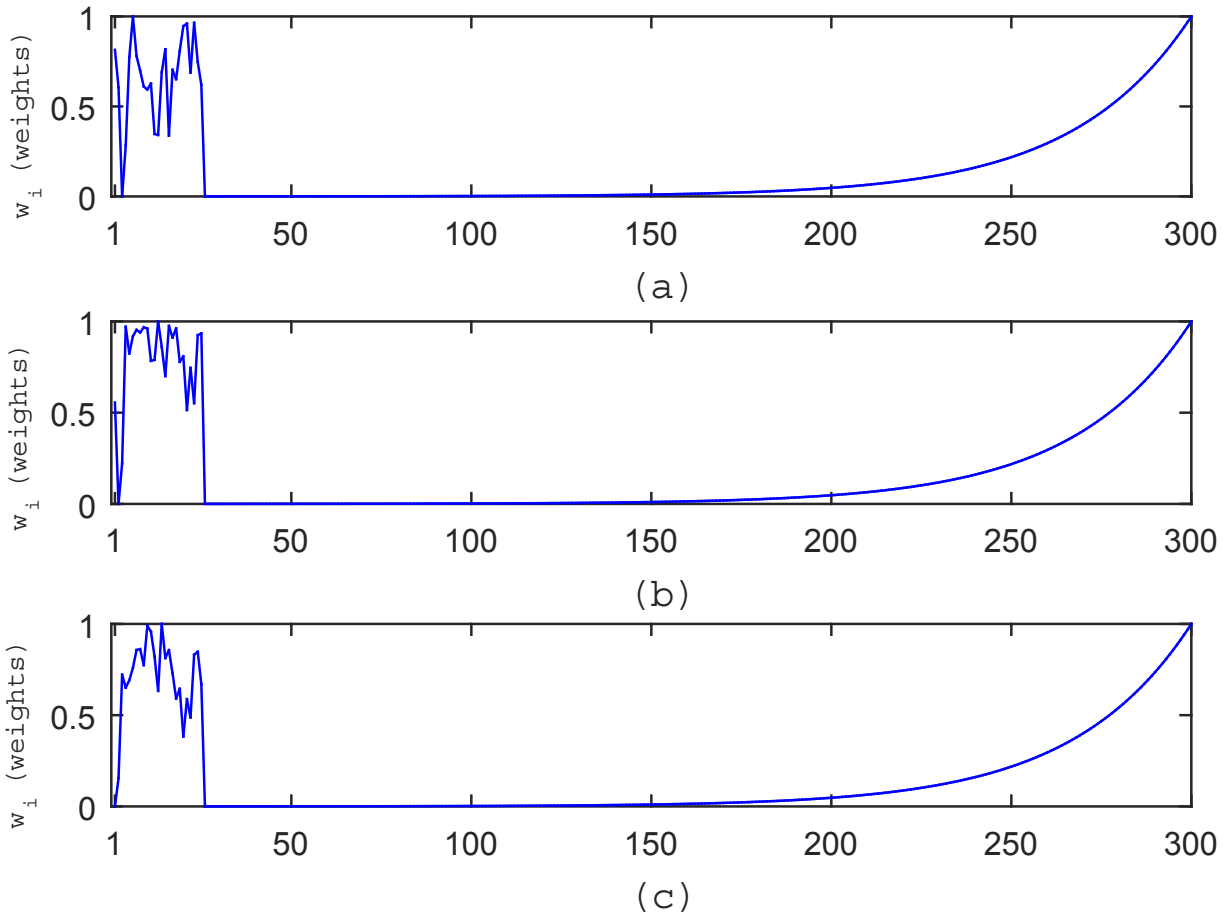

Figure 2.10: Figures (a), (b) & (c) display sample weights in RLWPLS for query 382, 383 & 384 respectively. 300:most recent & 1:oldest database sample.

2.5 Conclusion

In this chapter, the need for adaptation in soft sensors has been highlighted. Various mechanisms for adaptation in linear and nonlinear sensors are briefly discussed. Next, the JIT modeling framework and its steps and components are reviewed after which a new method for similarity calculation is proposed. The proposed method is able to deal with nonlinear and time varying systems simultaneously. Two popular local modeling approaches, LWLS and LWPLS, using the new similarity calculation are used under the JIT framework. Besides the typical offline method of parameter selection, it is shown that computationally efficient loocv using the PRESS statistic can be employed to calculate optimal parameters adaptively for every query. Finally the performance of the algorithms is evaluated on a numerical simulation and an industrial NIR data set. The results indicate that the proposed method of similarity calculation performs better than the traditional methods.

Chapter 3

Just-In-Time Modeling using Distance-Angle Similarity and Point-Based Bandwidth Estimation

3.1 Introduction

This chapter continues with the theme of JIT modeling explored in the previous chapter, i.e., with a focus on JIT based soft sensor modeling for prediction applications. Again, among the various constituents that make up a JIT model, the focus is on developing a new similarity criterion and method for bandwidth estimation of the local model involved. Having already discussed the JIT modeling steps and components at some length in Chapter 2, we skip these details here.

One of the most crucial elements of a JIT model is the similarity criteria used to select the subset of data for building the local model. As such, it is a key component on which the estimation accuracy of the model depends, and has therefore received a lot of research focus [14, 31]. This focus has led to the development of a number of similarity measures besides the traditional Euclidean distance (distance being inversely proportional to similarity), the distance (or space) and angle metric introduced by Cheng et al. [12] being one such criterion. This approach is applicable for dynamic systems and leverages the fact that for dynamic processes, the output not only depends on the current inputs but also on the evolution of the inputs. Hence, in a way, the angle and distance measure captures both the similarity of the inputs and also their evolution history.

The rest of the chapter is organized as follows. In Section 3.2, the distance-angle

similarity metric is briefly described and some shortcomings discussed. To address the highlighted issues, an improved distance-angle weight formulation is then proposed, followed by a discussion and comparison with the original form to highlight its advantages. Further, a novel point-based method to estimate the bandwidth/smoothing parameter of the JIT local model is also proposed. For this study we have considered time-invariant dynamic processes and Section 3.3 presents the application results followed by the concluding remarks in Section 3.4.

3.2 Space and angle weights

We now discuss the distance-angle similarity measure proposed in [12] to calculate database weights for dynamic systems. For a query, $\mathbf{x_q}$ (an m dimensional row vector), the similarity number, s_i, or the weight, w_i, of the i^{th} sample is given as:

$$w_i = \gamma\sqrt{e^{-{d_i}^2}} + (1-\gamma)\cos(\theta_i) \tag{3.1}$$

where d_i is the Euclidean distance, and θ_i is the the angle between the query, $\mathbf{x}_q$ and database sample, $\mathbf{x}_i$, calculated as:

$$\begin{aligned} \cos(\theta_i) &= \frac{\Delta\mathbf{x}_q {\Delta\mathbf{x}_i}^{\mathrm{T}}}{\parallel \Delta\mathbf{x}_q \parallel_2 . \parallel \Delta\mathbf{x}_i \parallel_2} \\ \Delta\mathbf{x}_q &= \mathbf{x}_q - \mathbf{x}_{q-1},\ \Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1} \end{aligned} \tag{3.2}$$

If $\cos(\theta_i) < 0$, the sample is discarded. γ is a balancing parameter between 0 & 1 that defines the relative importance of distance or angle in the similarity measure. When γ is 1, only distance measure defines similarity and for $\gamma = 0$ only angle measure is used. For all other γ values, a combination of distance and angle collectively determines similarity and hence the weight. Cheng et al. [12] propose carrying out an offline search over a number of γ values on the training set and the value that gives the least validation error is selected as the optimum one. This offline global approach to determine the parameters of a JIT model is computationally light and simple and is therefore commonly used. For convenience we denote the above mentioned modeling approach, i.e. the weight formulation in Eqn. (3.1) and the global offline method to determine the balancing parameter γ, as M1.

In the next section, we propose a new weight formulation to use angle information for calculating the sample weights. As will be seen later, this approach provides a better framework to assign sample weights and hence gives better results.

3.2.1 Angle as an additional variable

Since it is essential to the discussion that follows, we first define the role of the bandwidth/smoothing parameter in JIT modeling. The bandwidth/smoothing parameter determines the bandwidth or the region of validity of a local model. When the weight falls off very sharply with increase in distance, a very small subset of data is used to build the local model and the model is said to have a low bandwidth. Conversely, for a gradual decrease in weights with respect to increasing distance, a larger number of historical samples end up having high weights leading to a local model with a high bandwidth. Thus, the bandwidth in locally weighted models represents the tradeoff between robustness and prediction performance [14]. A local model with a high bandwidth is likely to give a biased but robust prediction whereas for a low bandwidth, the prediction will be less biased but with a higher variance. Choice of optimal value of the bandwidth/smoothing parameter depends on the system under study. Generally, for highly nonlinear systems or systems with low noise, a weight function with a steeper slope or short bandwidth is preferred. On the other hand, a weight function with a larger bandwidth is preferred for systems with a lesser degree of nonlinearity or high noise systems.

Given the above observation, the motivation to change the weight equation in (3.1) stems from the fact that $\cos(\theta)$ used in M1 is a specific weight curve for the angle similarity. As shown in Figure 3.1, it is a weight curve with a fixed bandwidth. This fixed bandwidth, therefore, does not allow freedom for the tradeoff between robustness and prediction performance mentioned earlier. To address the above shortcoming, we propose a new formulation based on the Gaussian kernel [2] to compute the sample weights.

$$\begin{aligned} w_i &= \mathrm{e}^{-\phi_s {d_i}^2}.\ \mathrm{e}^{-\phi_\theta \theta_i} \\ w_i &= w_s.\ w_\theta \end{aligned} \tag{3.3}$$

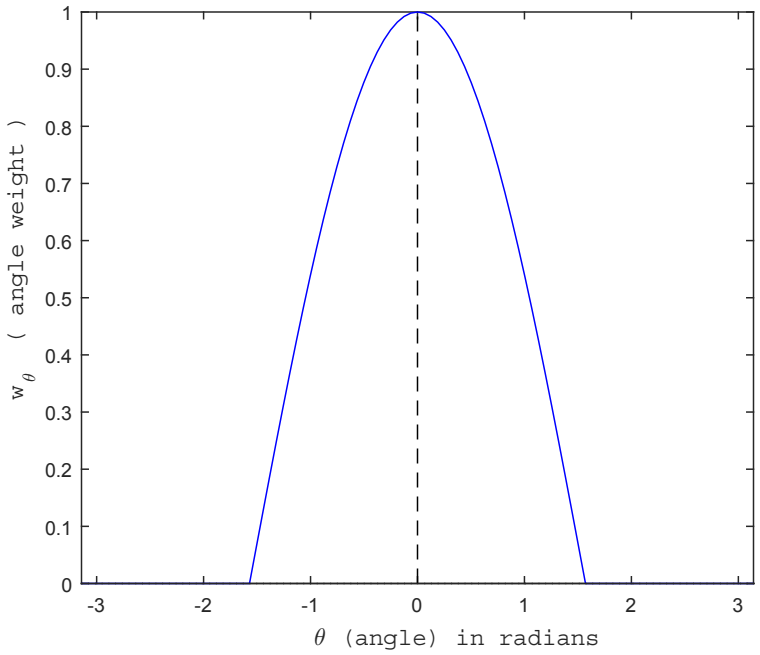

Figure 3.1: $\cos(\theta)$ weight function

where d_i and θ_i are the Euclidean distance and angle as calculated in Eqn. (3.2), and ϕ_s and ϕ_θ represent the smoothing parameters/bandwidths of the Euclidean space and angle respectively. The Euclidean distance is calculated after scaling the variables to avoid variables with large magnitude dominating distance calculations. Weighted Euclidean distance has also been used for calculating d_i [2], but since the focus here is on space distance and angle similarity as two different similarity criterion, we use the Euclidean distance with a single smoothing parameter.

θ can be thought of as an extra variable that contains information about the dynamics of the process. Thus, the formulation in Eqn. (3.3) is a more generalized representation incorporating angle information and appropriate setting of ϕ_θ will reduce Eqn. (3.3) to different cases. For example, $e^{-\phi_\theta \theta_i}$, can quite closely approximate $\cos(\theta_i)$ for $\phi_\theta = 0.8$, see Figure 3.2. The weight of any sample in Eqn. (3.3) is composed of two components, w_s and w_θ, the distance (or space) weight and the angle weight respectively. ϕ_s and ϕ_θ individually control how the two similarity criterion affect the weight calculations. For example, for $\phi_s = 0$ and $\phi_\theta = 0$, the regression becomes unweighted in both space and angle. As either of them increases, their

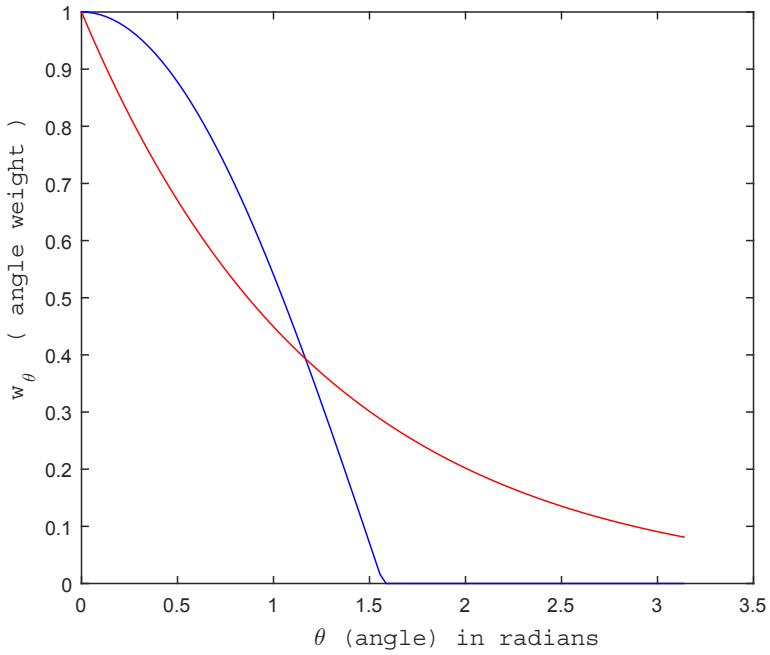

Figure 3.2: Weight functions comparison. - $\cos(\theta)$, - $e^{-0.8\theta}$

individual weight curves become steeper with respect to increase in space and angle respectively. We call JIT models using the weight formulation described in Eqn. (3.3) as M2. In terms of structure, it bears a close resemblance to the space-time similarity metric developed earlier in Chapter 2.

A comparison and analysis of the two weighting schemes in M1 and M2 is now carried out.

1. The weight calculation in Eqn. (3.3) is a generalized approach. By adjusting the ϕ_θ parameter, it is possible to adjust the bandwidth of the local model appropriately. For noisy systems, the optimum ϕ_θ is expected to be small whereas for systems with less noise, a large ϕ_θ is expected. It is worth noting that for the exponential weight function in Eqn. (3.3), ϕ is inversely proportional to the bandwidth. In the original approach however, the shape of the weight curve, and hence the bandwidth, with respect to angle, is fixed.

2. When it comes to the selection of the parameters, M2 has a more intuitive basis. At $\phi_s = \phi_\theta = 0$, neither space nor angle plays any role in weight calculations leading to unweighted regression. As either of them increases from 0, their

relative importance in the weight calculations rises. The selected values of ϕ_s and ϕ_θ should represent the optimum bias-variance tradeoff. On the other hand, in the case of M1, there is no such basis. As will be shown in the results section, there are situations when the training error is almost constant within a given range of values of the balancing parameter, γ. In such cases, there is no basis to favor γ selection towards either space or angle similarity.

3. Assuming angle as the dominant measure of similarity, i.e., γ & ϕ_s =0, the wight function in M1 goes to zero at a finite distance. Thus, points that are away from the query than that distance (where the weight drops to zero) can be ignored leading to faster implementations [2]. However, since in the case of M1 the bandwidth is fixed, there is always the risk that not enough points are present in the area with non zero weights. In the case of M2 however, the weight function has an infinite extent. Thus, it will be computationally heavy when database size is large since all samples will have non zero weights. This problem is easily addressed. In applications, generally the database size is kept fixed to some number and is not allowed to grow indefinitely. Alternately, one could impose a cut-off value so that samples with very low weights can be ignored without any error.

3.2.2 Bandwidth/Smoothing parameter estimation

Cheng et al. [12] use a global strategy to select the balancing parameter, γ, offline. In this section, we explore two strategies for estimating the bandwidth, i.e. the ϕ_s and ϕ_θ values. Both the strategies are applicable for time-invariant systems only. The first method is similar to the one used by Cheng et al. [12] and is based on a single global bandwidth that minimizes validation error on the training set. However, as demonstrated by Kim et al. [31], if the system has different degrees of nonlinearity at different operating points, then using a single bandwidth is not sufficient. Ideally, a low bandwidth (or high ϕ) should be selected at regions of high nonlinearity so that a small relevant neighborhood is selected. Similarly, for regions where the nonlinearity is not high, a high bandwidth should be selected so that maximum relevant points can take part in the local model and give a robust prediction. Figure 3.3, similar to

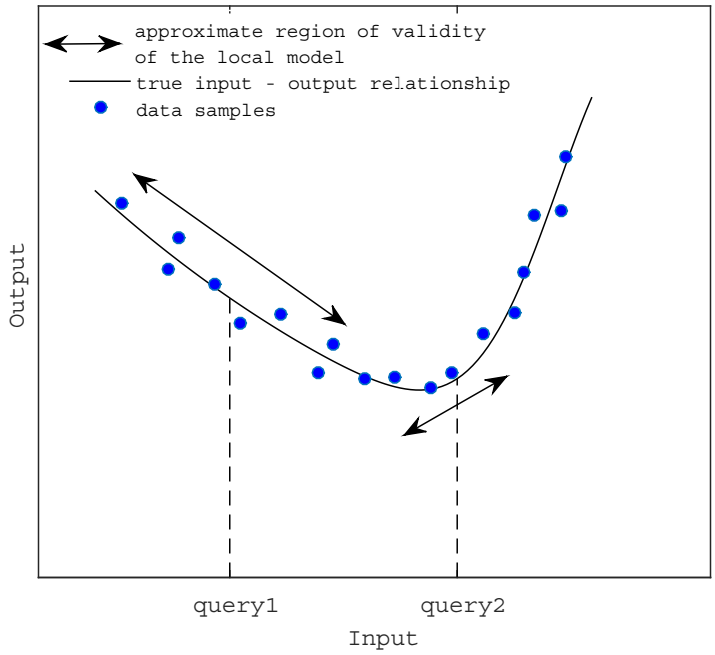

Figure 3.3: Variation of bandwidth with query

that used in [31], is generated to clearly illustrate and explain the concept.

In Figure 3.3, two queries, 1 & 2 are considered. Around query 1, the input-output relationship is relatively less nonlinear and hence a larger neighborhood should be selected. Using more points will reduce the affect of noise and increase prediction accuracy. For query 2 however, selecting a smaller neighborhood is more appropriate as the input-output relationship has a higher degree of nonlinearity. Thus, for the type of systems depicted in Figure 3.3, different bandwidth for different query variables should be used. The global strategy on the other hand leads to the selection of a single bandwidth that is a balance between all the local ones. This observation serves as the basis and motivation for the point-based local bandwidth estimation that is discussed next.

Point-based local bandwidth estimation

In this strategy each sample in the database is assigned a bandwidth associated with it [2]. Cross validation can be used to select the bandwidth parameter with respect to every database sample and stored with it. The steps for the bandwidth selection during the training phase and those during the prediction phase are described below.

Training Phase

Leave-one-out cross validation is used to select the bandwidth corresponding to any data point. Every sample in the database in turn acts as the query, $\mathbf{x_q}$. The ϕ_s, ϕ_θ pair that minimizes the prediction error for this query is then selected as the optimal pair associated with it. The following steps describe the training algorithm:
For $i = 1 : n$

$$\phi_{s_i}, \phi_{\theta_i} = \underset{\phi_s, \phi_\theta}{\arg\min} \mid y_i - \mathbf{x_i}\boldsymbol{\beta}\,\boldsymbol{(i)} \mid \tag{3.4}$$

$$\boldsymbol{\beta}\,\boldsymbol{(i)} = \Big(\mathbf{X}\,(\mathbf{i})^{\mathrm{T}}\mathbf{W}\,(\mathbf{i})\mathbf{X}\,(\mathbf{i})\Big)^{-1}\mathbf{X}\,(\mathbf{i})^{\mathrm{T}}\mathbf{W}\,(\mathbf{i})\mathbf{y}(\mathbf{i}) \tag{3.5}$$

$$\mathbf{W}\,(\mathbf{i}) = diag\,\{w_1, .., w_{i-1}, w_{i+1}, .., w_n\} \tag{3.6}$$

$$w_j = \mathrm{e}^{-\phi_s {d_j}^2}.\ \mathrm{e}^{-\phi_\theta \theta_j} \qquad j = 1, .., i-1, i+1, .., n \tag{3.7}$$

where the subscript i denotes the i^{th} observation and (i) denotes that this observation is not involved in calculations. Therefore, for example, y_i in Eqn. (3.4) denotes the i^{th} output and $y(i)$ in Eqn. (3.5) denotes all outputs except the i^{th}. The range of ϕ_s, ϕ_θ over which the minimization is performed depends on the system. High ϕ_s, ϕ_θ values can lead to overfitting and hence expert process knowledge, if available, should be used while selecting them [32]. ϕ_s, ϕ_θ values obtained from the global estimation method can be used as a guideline, and the range over which the minimization is performed can be selected around the global values. After the training phase is complete and a specific ϕ_s, ϕ_θ pair has been stored with each database sample, the prediction for a query variable is made as follows.

Prediction Phase

The prediction phase involves using a weighted approach to select the optimal ϕ_s, ϕ_θ pair for the given query. First, weights are calculated for every database sample with respect to the query and the weighted average of the ϕ_s, ϕ_θ values associated with the samples taken. However, since the initial weight calculation itself requires a ϕ_s, ϕ_θ

value, this leads to a Catch-22 situation. To avoid this, we use the ϕ_s, ϕ_θ value from the global estimation strategy to calculate sample weights for the initial step. For the time being, let us call the global smoothing parameter values as $\phi_{s_G}, \phi_{\theta_G}$. The following steps describe the prediction algorithm:
For query $\mathbf{x_q}$:

1. Calculate the initial database weights based on $\phi_{s_G}, \phi_{\theta_G}$:

$$w_i = \mathrm{e}^{-\phi_{s_G} {d_i}^2}.\ \mathrm{e}^{-\phi_{\theta_G}\theta_i} \tag{3.8}$$

$$\mathbf{W} = diag\left\{w_1, .., w_i, .., w_n\right\} \tag{3.9}$$

2. Calculate $\phi_{s_q}, \phi_{\theta_q}$ for the query, $\mathbf{x_q}$, using weighted average:

$$\phi_{s_q} = \frac{\sum\limits_i \phi_{s_i} w_i}{\sum\limits_i w_i} \quad , \quad \phi_{\theta_q} = \frac{\sum\limits_i \phi_{\theta_i} w_i}{\sum\limits_i w_i} \tag{3.10}$$

3. Re-calculate database weights using $\phi_{s_q}, \phi_{\theta_q}$:

$$w_i = \mathrm{e}^{-\phi_{s_q} {d_i}^2}.\ \mathrm{e}^{-\phi_{\theta_q}\theta_i} \tag{3.11}$$

$$\mathbf{W} = diag\left\{w_1, .., w_i, .., w_n\right\} \tag{3.12}$$

4. Make prediction, $\hat{y}_q$, using weighted regression with weights from Eqn. (3.12):

$$\boldsymbol{\beta_q} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{y} \tag{3.13}$$

$$\hat{y}_q = \mathbf{x_q}\boldsymbol{\beta_q} \tag{3.14}$$

3.3 Results and discussion

In this section, the different approaches are applied on a simulated and real life data set. In both cases, the system considered is time-invariant, i.e., the training/historical database is representative of all process characteristics. The general steps during prediction are:

1. Receive query, $\mathbf{x_q}$, from the test set
2. Calculate the weights of the database samples with respect to the query
3. Build local model (linear least squares) and make prediction corresponding to the query
4. Discard query and local model and return to step 1

Since we are considering time-invariant systems, the query does not represent new information and is not added into the database which is therefore kept fixed.

3.3.1 Numerical simulation

Here, the proposed methods are tested on a continuous stirred tank reactor (CSTR) simulation which has been used in literature for nonlinear system identification and control [33]. The coolant flow rate, q_c, and the product concentration of component A, C_A, respectively form the input, $\mathbf{x}$, and output, $\mathbf{y}$, of the irreversible, exothermic reaction. The following differential equations represent the CSTR dynamics [33].

$$\frac{dC_A(t)}{dt} = \frac{q(t)}{V}\left(C_{A0}(t) - C_A(t)\right) - k_0 C_A(t)\exp\left(\frac{-E}{RT(t)}\right) \tag{3.15}$$

$$\begin{aligned}\frac{dT(t)}{dt} =& \frac{q(t)}{V}\left(T_0(t) - T(t)\right) - \frac{(\Delta H)k_0 C_A(t)}{\rho C_p}\exp\left(\frac{-E}{RT(t)}\right) \\ &+ \frac{\rho_c C_{pc}}{\rho C_p V} q_c(t)\left\{1 - \exp\left(\frac{-hA}{q_c(t)\rho C_p}\right)\right\}\left(T_{c0}(t) - T(t)\right)\end{aligned} \tag{3.16}$$

An ARX model of order $(1, 2)$ is used to represent the SISO (single input single output) system.

$$y_t = a_1 y_{t-1} + b_1 u_{t-n_d} + b_2 u_{t-n_d-1} + \epsilon_t \tag{3.17}$$

where u_t is the input, $a_1,\ b_1$ & b_2 are the ARX model coefficients, $n_d = 1$ is the model delay & ϵ_t is white noise which is artificially added to the data set. Two cases, with and without noise, are considered to illustrate the role of the smoothing parameters. 500 samples are generated for the training and test sets respectively. The input is generated by a random uniform distribution in the interval $[101, 103]$ with a switching probability of 0.3 at every sampling time. Figure 3.4 shows the input-output data generated with no additive noise. The following methods are tested:

1. M1: Existing JIT approach with space and angle weights. Single global γ is obtained offline by minimizing the leave-one-out cross validation error on the training set. Every sample is removed once from the training set and the remaining samples are used to predict its value. The γ value that minimizes the rmsep is selected as the global optimum.

2. M2: Proposed JIT approach with space and angle weights. Global smoothing parameters, ϕ_s and ϕ_θ, are obtained as in M1 above.

3. M3: Proposed JIT approach with space and angle weights with point-based local smoothing parameters. Estimation of the bandwidth and prediction is carried out as explained in Section 3.2.2 earlier.

Case I: $\epsilon_t \sim N(0, 0.001^2)$

Figure 3.5 displays the noisy vs. clean output plot for the training set and Figures 3.6 & 3.7 show the training results for models M1 and M2 respectively. In Figure 3.7, the rmsep value is plotted against different ϕ_s and ϕ_θ pairs. $\phi_s = [0.0, 0.1, 0.2, 0.3, 0.4]$ and $\phi_\theta = [0, 1, 2, 3, ..., 9]$ giving a total of 50 pairs. Each stem plot within the black grid lines displays the rmsep variation with ϕ_θ (shown on the bottom x axis) for a fixed value of ϕ_s (shown on the top x axis). The rmsep value decreases very slightly with respect to ϕ_s initially and then increases (not evident from the figure). Hence, it is reasonable to pick $\phi_s = 0$ as the optimum value. With respect to ϕ_θ, the minimum

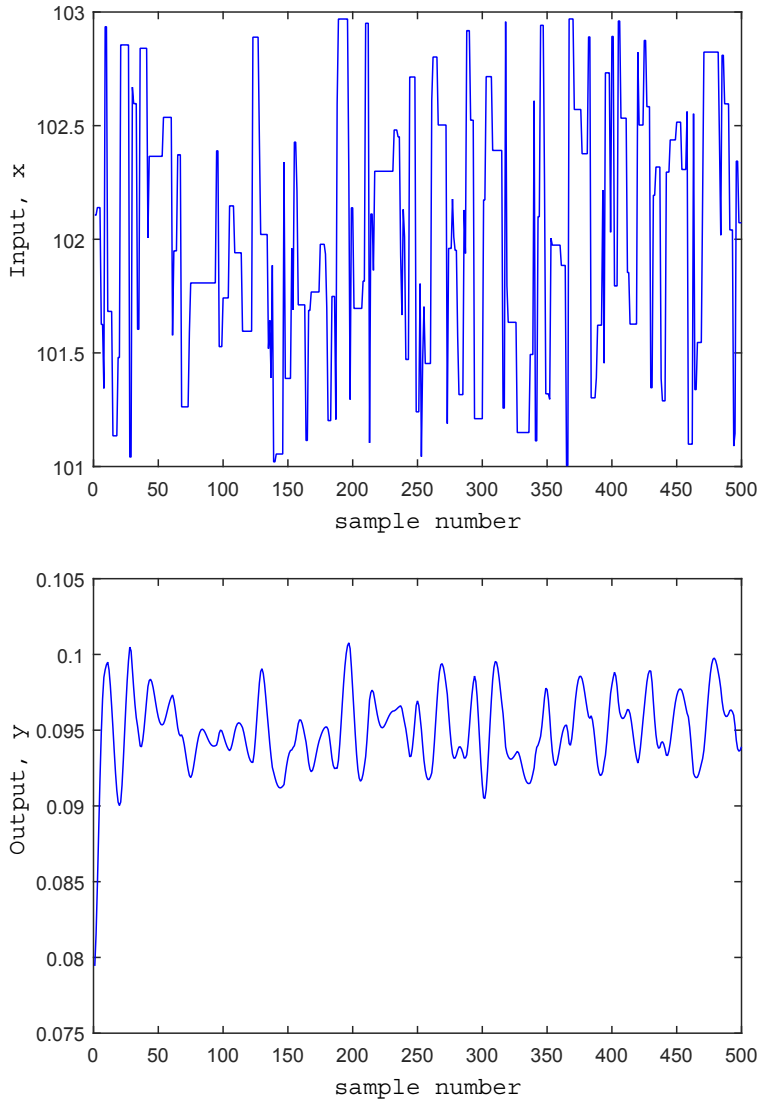

Figure 3.4: Input-Output data

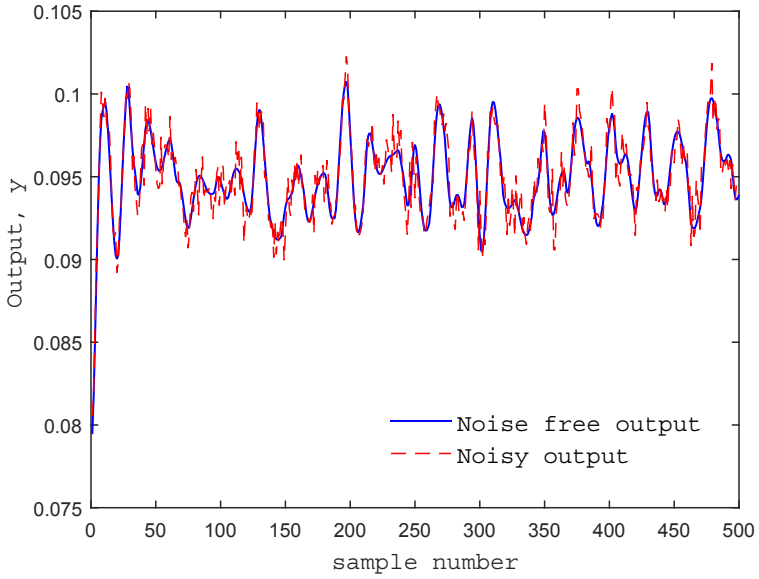

Figure 3.5: Noisy vs. Clean output

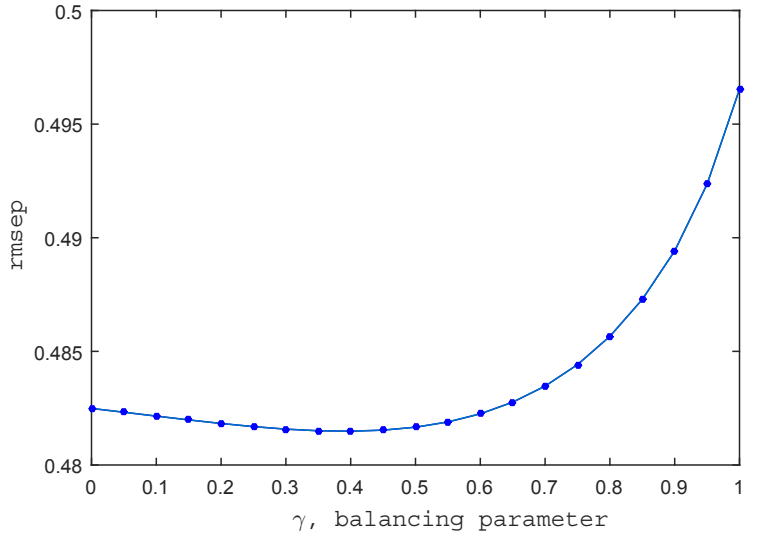

Figure 3.6: M1 training result, Case I

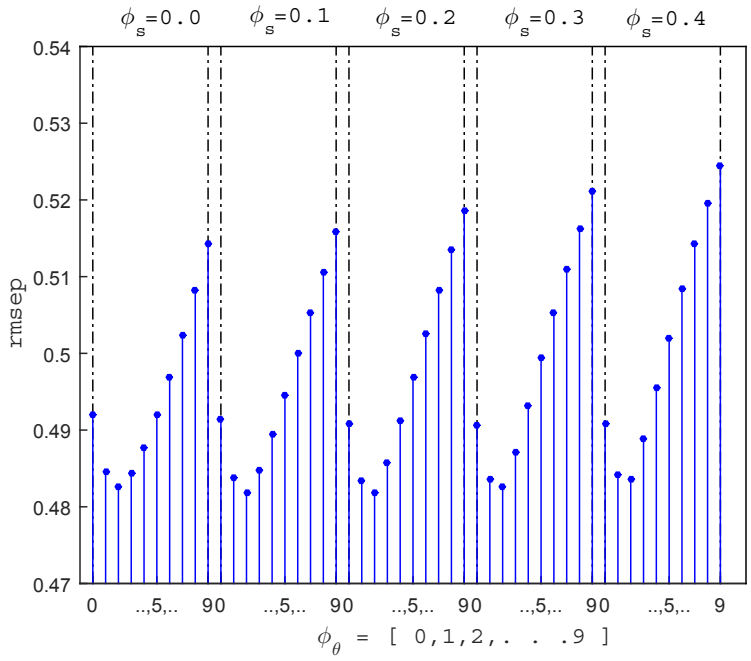

Figure 3.7: M2 training result, Case I

error is obtained at $\phi_\theta = 2$ but 1 is selected as the optimum. As discussed earlier, a higher ϕ tends to lead to a noisy estimate and hence, if an increase in the value of the smoothing parameters does not lead to appreciable decrease in rmsep, the lower value is selected. Thus, the optimum $\{\phi_s, \phi_\theta\}$ pair is $\{0, 1\}$, implying that the prediction is unweighted with respect to distance in space. From the M1 training result in Figure 3.6, it is observed that minimum rmsep is obtained around $\gamma = 0.40$. We make the following observations regarding the training results:

1. Angle similarity is dominant in comparison to space similarity.

2. The selection of ϕ_s and ϕ_θ in the case of M2 is based on the bias-variance tradeoff as discussed earlier. However, in the case of M1, although the minimum rmsep value occurs at $\gamma = 0.4$, its value is almost the same for $\gamma = 0.5$ to 0.0. There is no guideline for favoring either space or angle similarity in such cases.

3. The training results show that rmsep decreases slightly with increase in ϕ_θ till around 2 and that space similarity has a minor role. Hence, the range of ϕ_s and ϕ_θ for M3, based on the training results obtained for M2, is:

   $\phi_s = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$

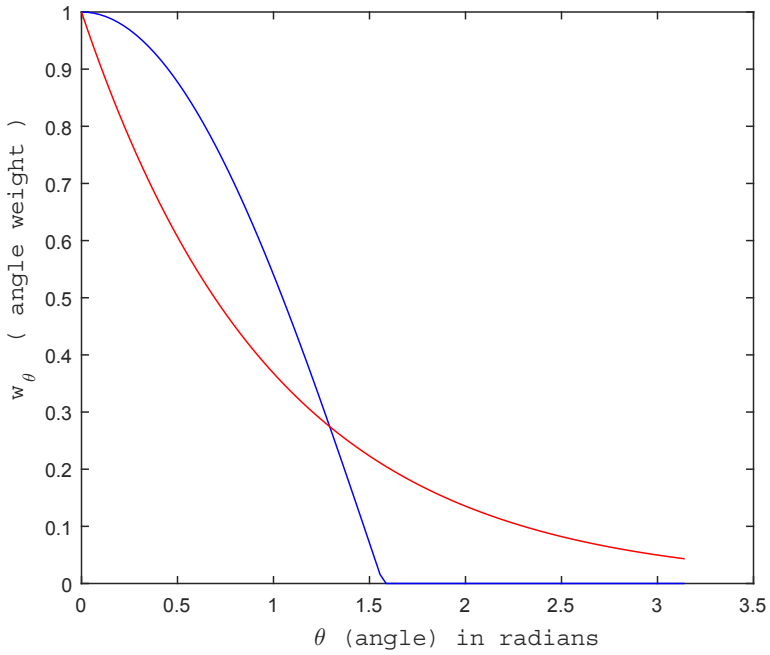

Figure 3.8: Angle weight curves of M1 & M2, Case I. - $\cos(\theta)$, - $e^{-\theta}$

$\phi_\theta = \{0, 0.4, 0.8, 1.2, 1.6, 2\}$

Table 3.1 shows the results on the test set.

Table 3.1: Test Results: Numerical simulation - Case I

Model	γ	ϕ_s	ϕ_θ	R	rmsep
M1	0.40	-	-	0.83	0.516
M2	-	0	1	0.83	0.517
M3	-	-	-	0.83	0.517

For this case the performance of all three models is nearly the same. Although the weight curve for M1 is not exactly the same as the cos function weight curve, it is very similar (since the balancing parameter in favor of angle similarity is 0.60). Figure 3.8 shows that the angle weight curves of M1 and M2 are similar and this is reflected in the near identical performance shown in Table 3.1. The query-wise plots of ϕ_s & ϕ_θ for model M3 are shown in Figure 3.9. We can see that the ϕ values for M3 are concentrated around the global value obtained for M2 which results in a similar performance. A possible reason is that the variation in the degree of nonlinearity at different operating regions is not high and is masked by the presence of noise.

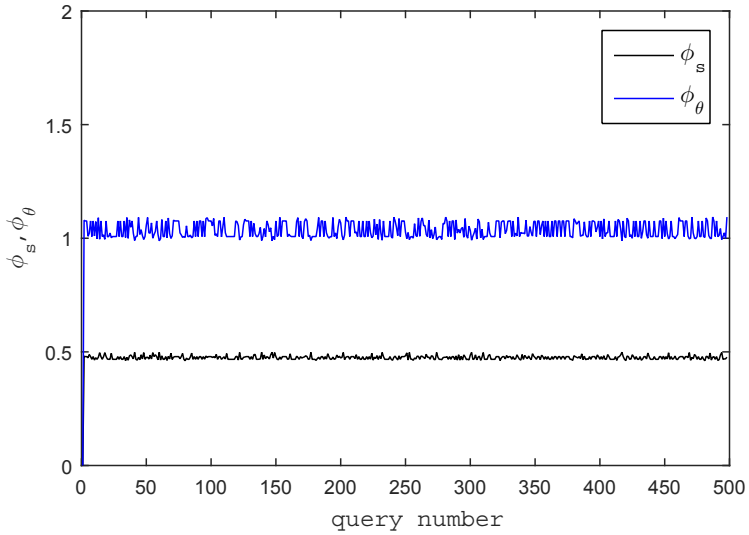

Figure 3.9: Query-wise ϕ_s & ϕ_θ values for M3, Case I

Case II: $\epsilon_t = 0$

For Case II, no noise is added to the system. Figures 3.10 and 3.11 show the training results for M1 & M2. We can see from the training results that the optimum value of γ for M1 is 0.65. Again, for M2, a similar figure as in Case I earlier is obtained. Following similar guidelines for selection of ϕ_s, ϕ_θ, the optimum pair for M2 in Case II is $\{0, 6\}$. One can also note that for M2, there is a slight decrease in rmsep initially with respect to ϕ_s which is consistent with the training result of Case I. The following points are noted:

1. From the training results of model M2, angle similarity seems dominant whereas space similarity seems dominant in the case of M1 (γ value of 0.65). The only difference between Cases I & II is the addition of noise. As such, the system properties are not affected in any way. Also, since Case II has zero noise, it is expected that selecting a smaller relevant subset for local modeling should increase prediction accuracy. M2 results are consistent with this observation. ϕ_s in Case I & Case II is nearly zero indicating that similarity calculations are unweighted with respect to distance in space. ϕ_θ increases from 1 to 6 indicating a decrease in bandwidth and hence a smaller neighborhood selection which, given the same system, indicates a decrease in noise. Figure 3.12 shows

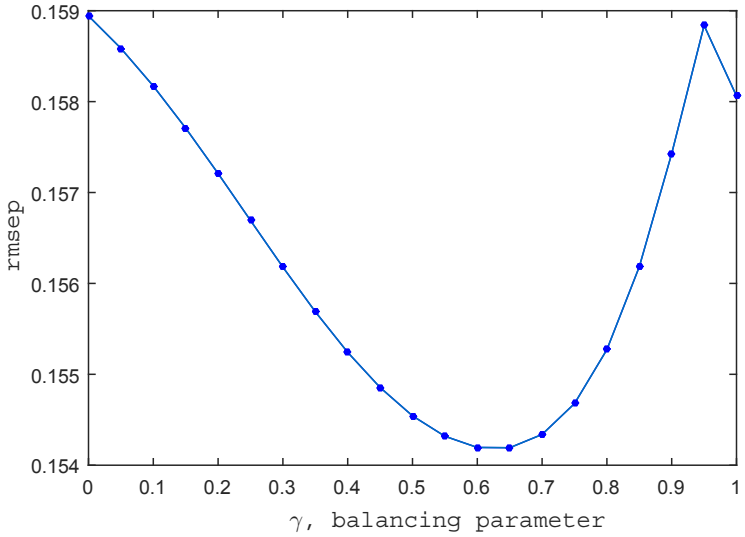

Figure 3.10: M1 training result, Case II

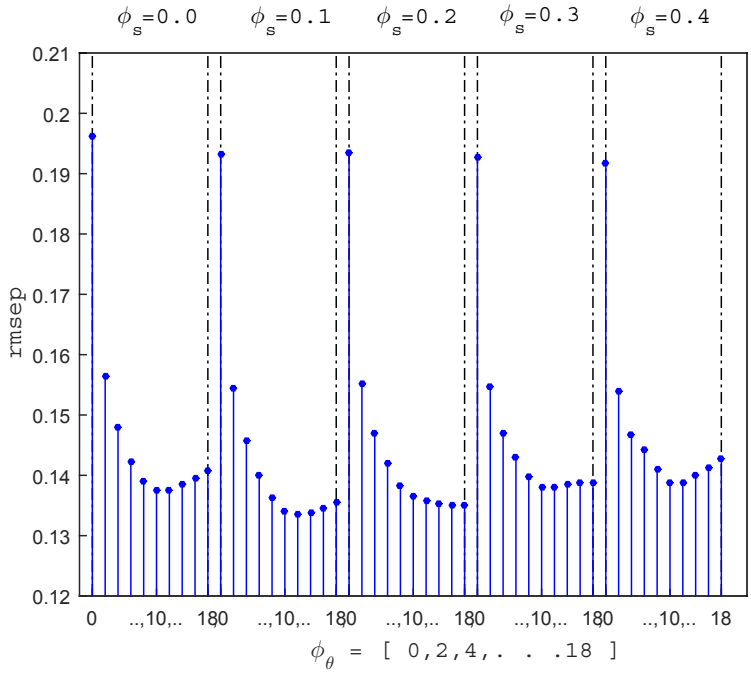

Figure 3.11: M2 training result, Case II

the weight curves for M2 for the 2 cases. M1 on the other hand does not give results consistent with the above observations. The influence of noise causes a shift in the optimum balancing parameter value from 0.40 in Case I to 0.65 in Case II. Although, this causes improvement in M1 prediction results from Case I to Case II, the results are hard to interpret. Without prior knowledge of the fact that Case I is simply Case II with additional noise, inferring this from M1 training results is not possible. This is because the $\cos(\theta)$ weight function used in M1 is fixed and is unable to adjust to the changing noise levels. Overall, this is reflected in a better performance of M2 over M1, not only in terms of prediction accuracy but also in terms of interpretation of the results.

2. Similar to Case I, the selection of ϕ_s and ϕ_θ for M2 is based on the bias-variance tradeoff. For M1 however, although the minimum rmsep value occurs at $\gamma = 0.65$, it remains almost unchanged for $\gamma = 0.5$ to 0.75.

3. For M2, the training results show that rmsep decreases slightly with increase in ϕ_θ till around 12 and that space similarity has a minor role. Hence, the range of ϕ_s and ϕ_θ for M3, based on the training results obtained for M2, is:

   $\phi_s = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$

   $\phi_\theta = \{0, 2, 4, 6, 8, 10, 12\}$

Table 3.2 shows the results on the test set. Clearly, M2 is able to adjust the angle weight curve to the lower noise in Case II and give a more relevant subset for local modeling than M1. The effect of this is that the prediction by M2 is more accurate than M1. The performance of M3 is better than either M1 or M2. Figure 3.13, which displays the query-wise plots of ϕ_s & ϕ_θ for M3, helps explain this.

Table 3.2: Test Results: Numerical simulation - Case II

Model	γ	ϕ_s	ϕ_θ	R	rmsep
M1	0.65	-	-	0.986	0.150
M2	-	0	6	0.988	0.139
M3	-	-	-	0.990	0.126

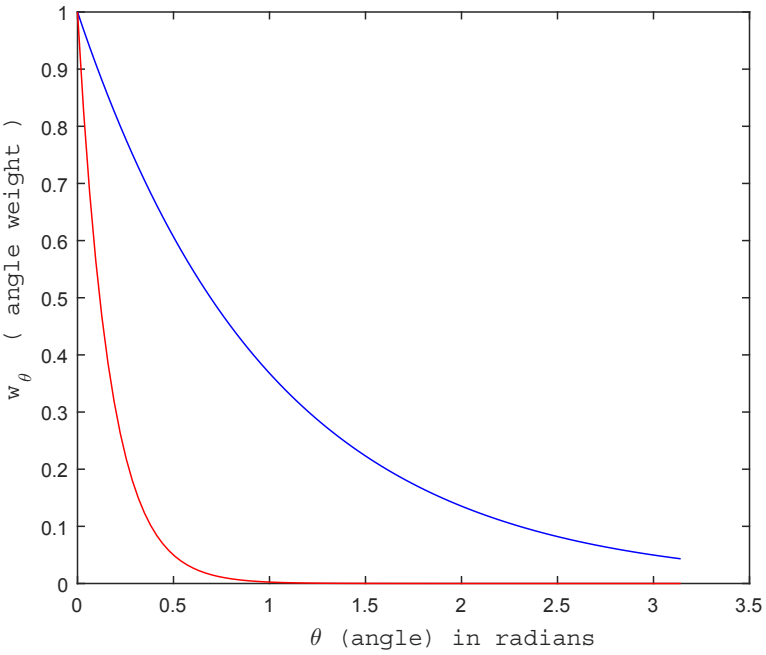

Figure 3.12: M2 weight functions comparison for Case I & Case II. - $e^{-\theta}$, - $e^{-6\theta}$

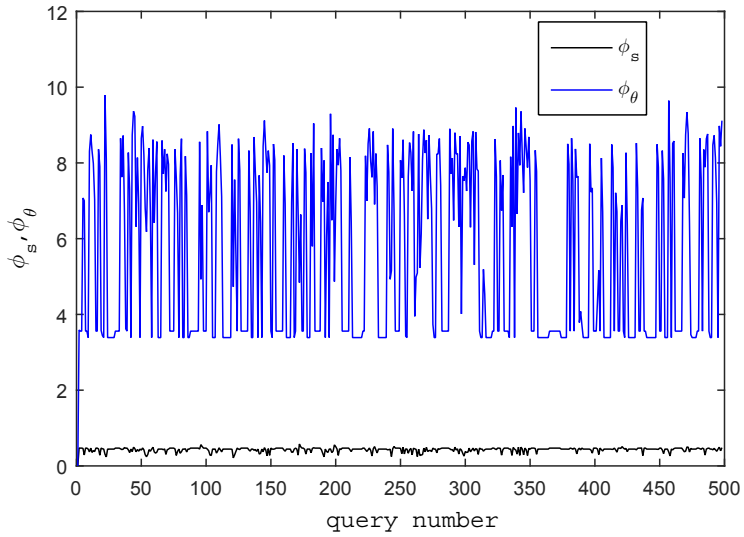

Figure 3.13: Query-wise ϕ_s & ϕ_θ values for M3, Case II

A larger variation in ϕ_θ values is observed around the global value of 6 which was obtained for M2. This is reflected in a better prediction performance as displayed in Figure 3.14, where a section of the prediction results is plotted against the reference values.

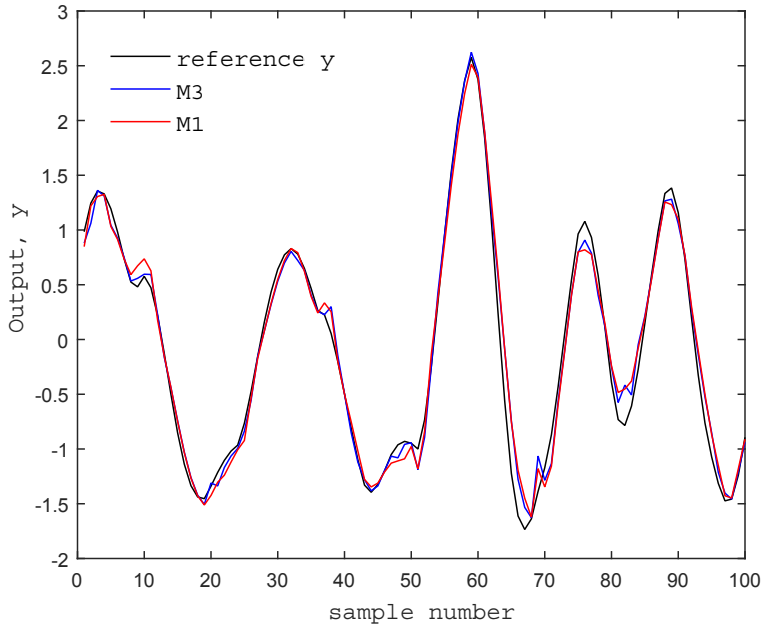

Figure 3.14: M1 vs M3 prediction comparison

3.3.2 Experimental case study

In this section, the methods are applied to data obtained from an experimental setup. The system studied is the 4-tank system and Figure 3.15 shows the schematic [34]. There are 2 inputs and 2 outputs forming a MIMO (multiple input multiple output) system.

Outputs: **y1** and **y2**

y1: output1, water level of bottom left tank, metres

y2: output2, water level of bottom right tank, metres

Inputs: **x1** and **x2**

x1: flow rate of left pump, litre/min

x2: flow rate of right pump, litre/min

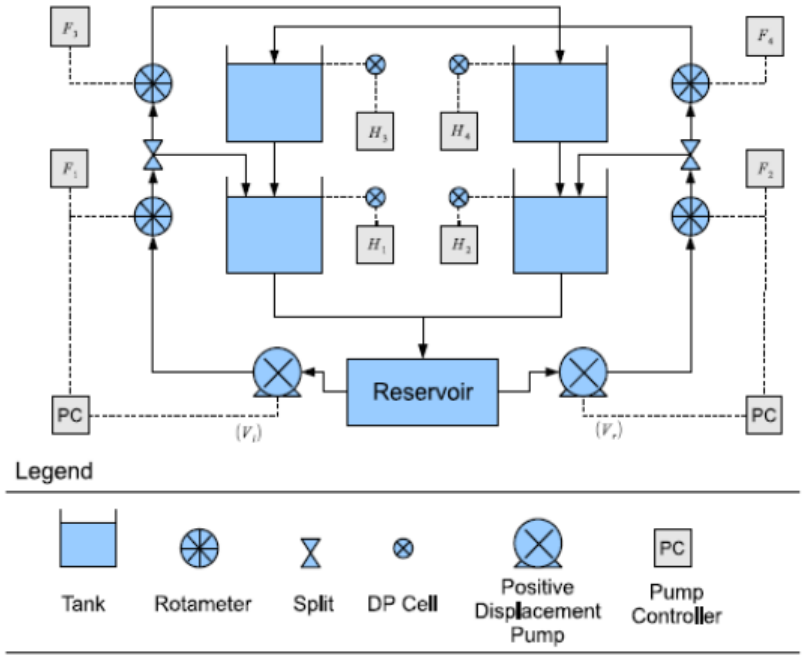

Figure 3.15: Schematic of Four-Tank System

Based on the first principles model, it is known that the system is nonlinear, and because of symmetry, there are two identical single output multiple input systems formed by $\mathbf{y1}, \mathbf{x1}, \mathbf{x2}$ and $\mathbf{y2}, \mathbf{x1}, \mathbf{x2}$ respectively. Figure 3.16 shows the plots of the two outputs. Although the two systems are identical, Figure 3.16 shows that output **y2** is noisier than **y1**. One should then expect results similar to the ones obtained for the numerical simulation in Section 3.3.1. An ARX model of order $(2, 2)$ with delay as $[1, 2]$ & $[4, 1]$ is used as the local model structure in the JIT framework. A total of nearly 1100 samples are obtained with a random binary signal being used as the input signal for sample generation. The first 700 samples form the training set, and the remaining 400 form the test set and the models M1, M2 and M3 described previously are tested.

Case I: Output y2

For Case I, the output considered is **y2**. Figures 3.17 & 3.18 display the training results. From Figure 3.17, it can be seen that a lower training error is obtained for low γ values in the case of M1 indicating that angle similarity is dominant while

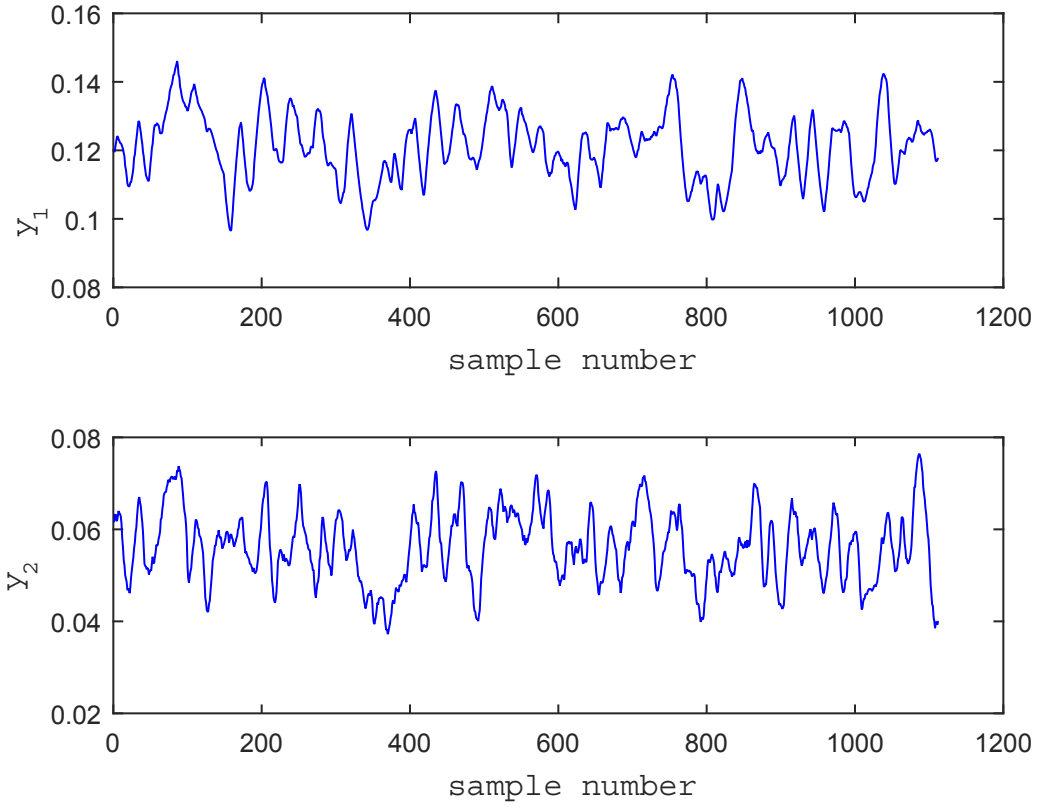

Figure 3.16: Output plots of the four tank system

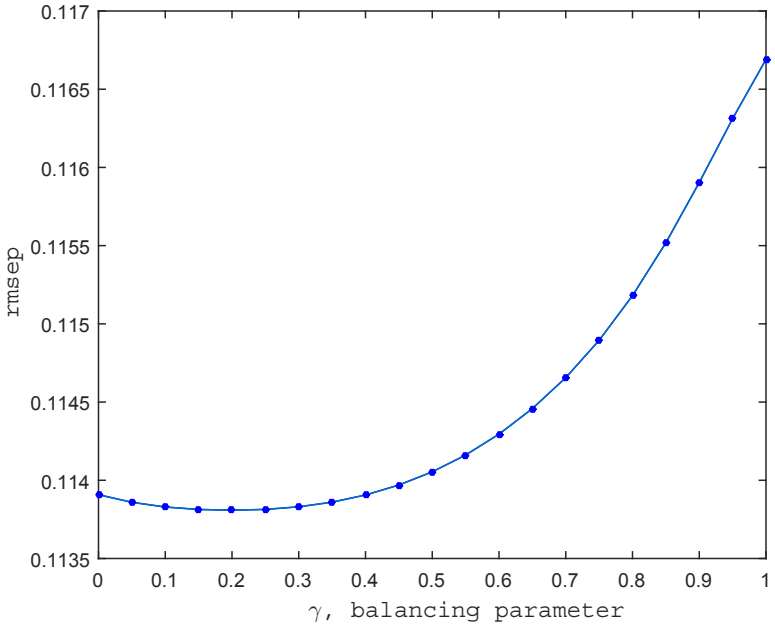

Figure 3.17: M1 training result, Case I

space similarity plays a minor role. The minimum error is obtained for $\gamma = 0.2$ which is selected as the optimum. In the case of M2, the space weights do not cause any reduction in training error. 0 & 2 are therefore selected as the optimum ϕ_s & ϕ_θ values respectively for M2. For M3, based on Figure 3.18, the following range of ϕ_s, ϕ_θ is selected:

$\phi_s = \{0\}$

$\phi_\theta = \{0, 0.4, 0.8, 1.2, 1.6, 2\}$

Table 3.3 presents the test set results. All models give very similar results. As before in Case I of Section 3.3.1, the weight curve of M2 with $\phi_s = 0$ & $\phi_\theta = 2$ is quite similar to that of M1 with $\gamma = 0.2$. In the case of M3, the query-wise plot of ϕ_θ, shown in Figure 3.19, indicates that the ϕ_θ values are concentrated around the value of 1 which is why the performance of M3 is similar to that of M1 and M2.

Table 3.3: Test Results: Experimental case study - Case I

Model	γ	ϕ_s	ϕ_θ	R	rmsep
M1	0.2	-	-	0.99	0.104
M2	-	0	2	0.99	0.103
M3	-	-	-	0.99	0.103

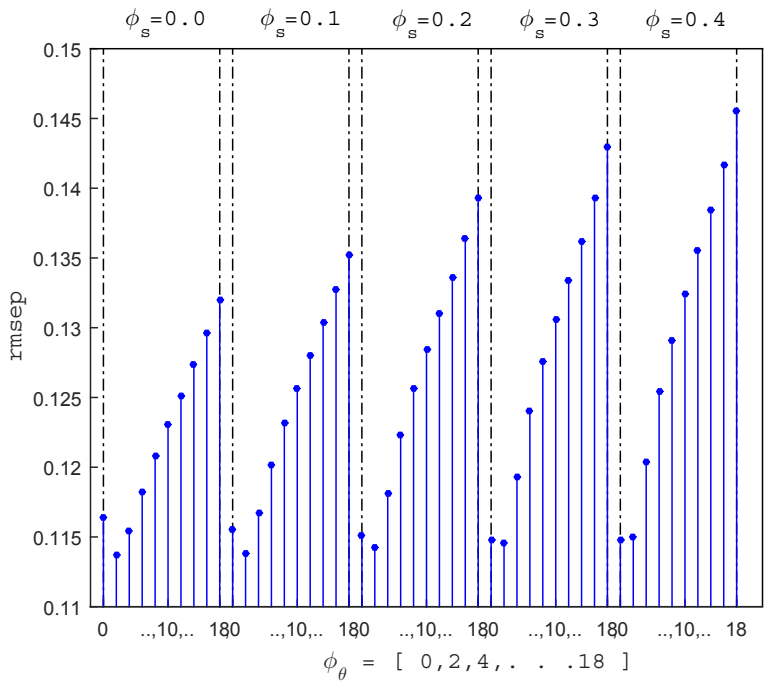

Figure 3.18: M2 training result, Case I

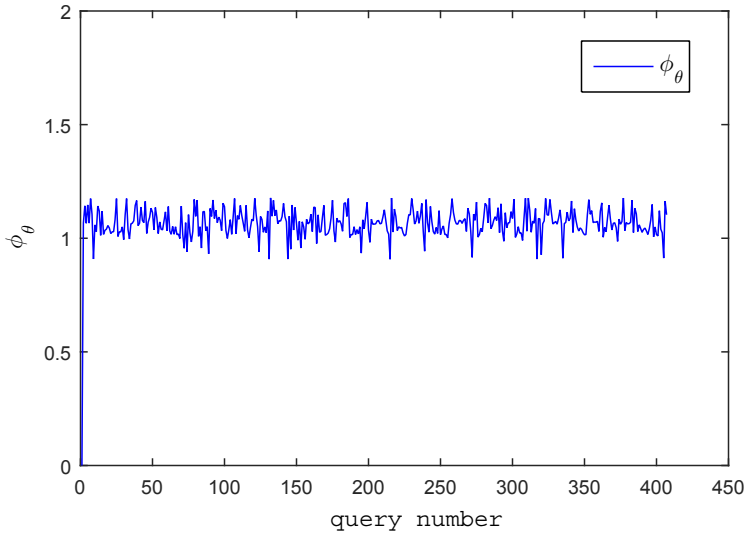

Figure 3.19: Query-wise ϕ_θ values for M3, Case I

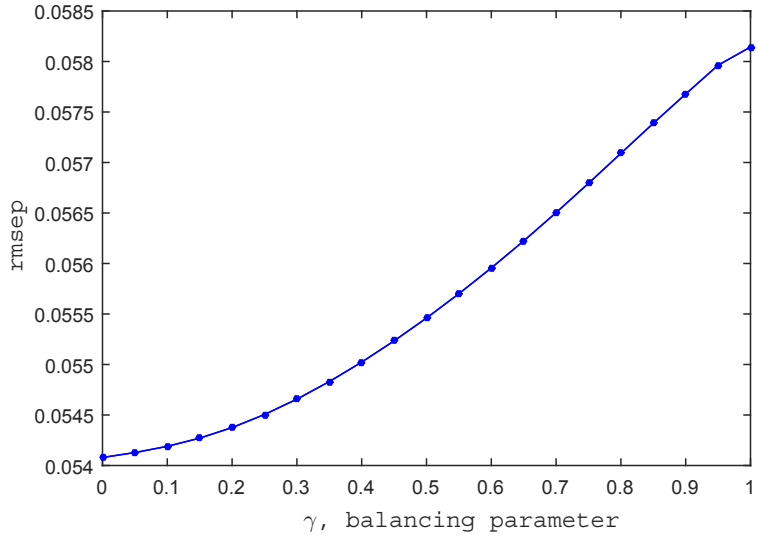

Figure 3.20: M1 training result, Case II

Case II: Output y1

For Case II, **y1** is the predicted output variable. It was mentioned before that the systems associated with the two outputs, **y1** & **y2**, are symmetric and the only difference is that **y2** is contaminated by more noise than **y1**. Therefore, one should expect better performance in this case and the model parameters should also reflect this change in noise level. Figures 3.20 & 3.21 display the training results for models M1 and M2 respectively.

For M1, the optimum γ value of 0, which is evident from Figure 3.20, is not consistent with its value of 0.2 in Case I. For M2, $\{0, 4\}$ is selected as the optimum ϕ_s, ϕ_θ pair, which is consistent with the results of Case I earlier. ϕ_s remains constant at 0 and the increase in ϕ_θ from 2 in Case I to 4 in Case II reflects the decrease in noise in the output, **y**, from Case I to Case II. For M3, the range of ϕ_s, ϕ_θ values selected is:

$\phi_s = \{0\}$

$\phi_\theta = \{0, 1, 2, 3, 4, 5, 6\}$

Table 3.4 shows the results on the test set.

As expected, both M2 and M3 perform better than M1 and all 3 models perform better than in Case I. From the query-wise plot of ϕ_θ in Figure 3.22, it is observed that the ϕ_θ values for M3 do not have a large spread and are close to the global value

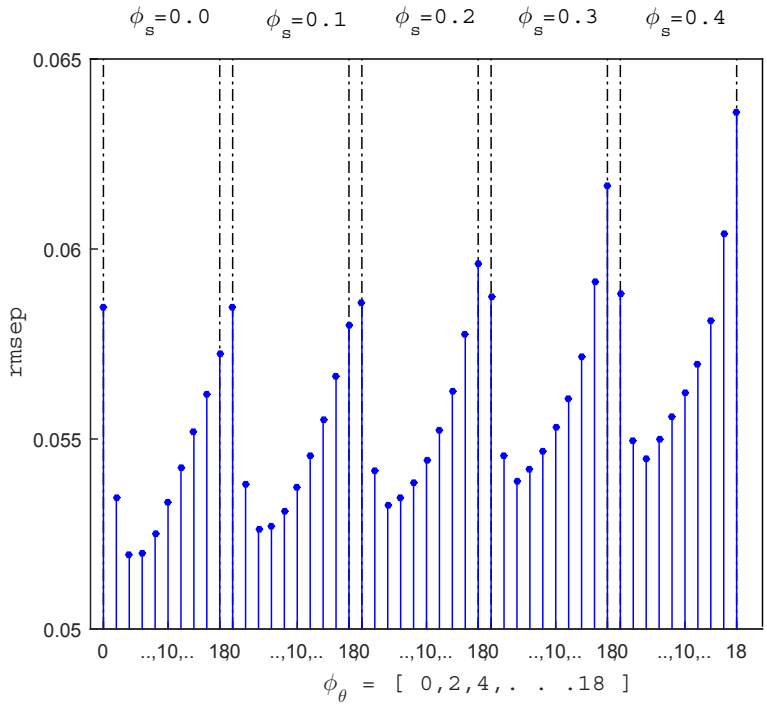

Figure 3.21: M2 training result, Case II

Table 3.4: Test Results: Experimental case study - Case II

Model	γ	ϕ_s	ϕ_θ	R	rmsep
M1	0.0	-	-	0.99	0.050
M2	-	0	4	0.99	0.047
M3	-	-	-	0.99	0.048

of 4 obtained for M2. This indicates that the degree of nonlinearity for this system stays more or less constant and is the reason why M3 does not perform better than M2 unlike in the CSTR simulation of Section 3.3.1.

3.4 Conclusion

The chapter begins with an introduction to the space (or distance) and angle based similarity metric used to calculate sample weights for JIT modeling of dynamic processes. A discussion of the role of the bandwidth in local models serves to point out the shortcomings of the metric and provides the motivation for an alternate approach. A new weight formulation using distance and angle is therefore developed to address the issues highlighted. Further, a novel point-based, local bandwidth estimation method is proposed. At an increased computation cost during offline training,

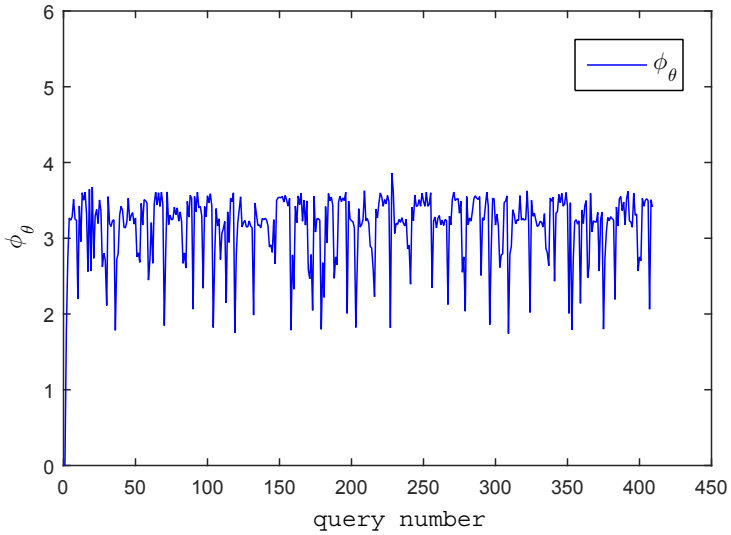

Figure 3.22: Query-wise ϕ_θ values for M3, Case II

the point-based method determines query specific bandwidth as opposed to a single bandwidth obtained from the global strategy. The advantages of the methods are demonstrated by application to a CSTR simulation and experimental data.

Chapter 4

Adaptive Linear Regression: Cautious and Robust Parameter Update Strategy*

4.1 Introduction

Mathematical models are implemented in a variety of industries like the steel, refinery and pharmaceutical industries for the purpose of process control, product quality estimation or fault detection. The purpose of these models, also called soft sensors, is to estimate process variables that are either not possible to measure using hardware analyzers or are too expensive to measure [4]. In the presence of sufficient process knowledge, white box models which are based on first principles are used for these mathematical models. First principle models require detailed knowledge about the physical and chemical phenomena underlying the processes and typically describe the ideal or desired state of operation of the process plant [5]. However, due to the complexity of industrial scale processes and influence of external factors, this is rarely the case and the necessary information required for white box models is generally not available. In such cases, black box modeling is a suitable alternative. Black box models are data driven models and can be built solely on the available data. Grey-box modeling is a middle path which combines available process know-how with the data driven modeling technique to build useful mathematical models [5].

As discussed in earlier chapters, among data based approaches, global modeling,

*This chapter is an extended version of the under review paper: Shekhar Sharma, Swanand Khare, and Biao Huang. Robust Online Algorithm for Adaptive Linear Regression Parameter Estimation and Prediction. *Journal of Chemometrics* (submitted, 2015).

also called offline or batch modeling, has been the traditional choice [7]. However, models such as artificial neural network (ANN), built using historical data have certain disadvantages.

- First, if the historical database contains operation in multiple operating modes then use of a single offline model for all the operating modes is inefficient.
- Second, for satisfactory performance of the model, the historical database should contain all possible operating states and conditions, which is generally not the case due to the time varying nature of industrial processes

Typical causes of such time-varying behavior are [7]:

1. changes of process input (raw) materials;
2. process fouling;
3. abrasion of mechanic components;
4. changes in catalyst activities;
5. production of different product quality grades;
6. changes in external environment (e.g. weather, seasons).

These points highlight the need for model maintenance and re-training of model parameters. Hence, equipping soft sensors with some adaptation strategy has become almost a necessity. The adaptation or re-training of complex nonlinear models like artificial neural networks or neuro fuzzy networks is not convenient [7]. The complexity and high computation load associated with such models also limits their applications. On the other hand, adaptive versions of linear models are simple and easy to implement, thus making them a popular choice for soft sensor applications. Blockwise linear least squares or moving/sliding window least squares (MWLS) and recursive least squares (RLS) are two such popular adaptive linear regression techniques. For high dimensional and or highly correlated data, moving window and recursive versions of principal component analysis (PCA) and partial least squares (PLS), which

are essentially linear techniques, have found wide applications. In this chapter however, we concentrate on adaptive linear regression techniques where dimensionality reduction is not a requirement and focus on soft sensor applications, specifically, on online prediction and parameter estimation in the presence of unknown process drifts or parameter changes.

Least squares cost function based regression techniques, though widely used, are not robust. Robustness refers to the insensitivity of the estimates produced by a method to outlying or abnormal data points or model misspecifications [35]. Least squares estimates are optimal and coincide with the maximum likelihood estimate under the assumption of independent and normally distributed error terms [35, 36, 37]. However, in practical scenarios, this assumption does not always hold and just a single outlying observation can distort the least squares estimate [37]. Therefore, robust alternative regression techniques are needed. L_1 or least absolute deviation (LAD) and the online passive aggressive algorithm (OPAA) [38] are two such adaptive linear regression methods, though they are not as widely used. As will be shown later in the chapter, both LAD and OPAA in their current forms have certain disadvantages when it comes to practical applications. In this chapter, we propose a new algorithm called Smoothed Passive Aggressive Algorithm (SPAA) which overcomes some of these shortcomings making it suitable for practical implementation, though at the expense of additional computational cost. SPAA is not only robust but also follows a cautious parameter update strategy and is not influenced by minor disturbances by skipping the parameter update step. Further, it will be shown that the proposed SPAA framework is a general one, and for specific values of the tuning parameters, it reduces to OPAA or a moving window version of the LAD algorithm.

The rest of the chapter is organized as follows. Section 4.2 describes some of the popular adaptive linear regression techniques. Section 4.3 begins with a comparison of these techniques followed by the new proposed algorithm. The application results and discussions are presented in Section 4.4 and finally, the chapter closes with the concluding remarks in Section 4.5.

4.2 Adaptive linear regression

Ordinary least squares is one of the most popular linear regression techniques. However, for system identification and parameter estimation in the presence of unknown parameter changes, its adaptive versions, namely the sliding or moving window and recursive least squares have been widely used [39]. Another less known, from industrial application point of view, adaptive linear regression technique is OPAA. A brief overview of these algorithms is provided in the sections below.

4.2.1 Recursive least squares (RLS)

Consider the case where the relation between the output $\mathbf{y}$ (or difficult-to-measure variable) and the input $\mathbf{X}$ (or easy-to-measure variables) can be reasonably approximated by the following linear form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{4.1}$$

where $\boldsymbol{\epsilon}$, the residual vector, consists of independent and Gaussian distributed entries. Next, let us assume that n number of observations have been accumulated. The least squares (LS) estimate of the regression vector, $\hat{\boldsymbol{\beta}}$ can be obtained as [40, 41]:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{4.2}$$

For the case of online prediction applications, new observations are continuously available. Consider the next available observation, y_{n+1} & $\mathbf{x}_{n+1}$. This new information can be incorporated into the old one and a new estimate of the regression vector, $\hat{\boldsymbol{\beta}}_{n+1}$, can then be obtained as follows [40]:

$$\begin{bmatrix}\mathbf{y}_n \\ y_{n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_n \\ \mathbf{x}_{n+1}\end{bmatrix}\boldsymbol{\beta}_{n+1} + \begin{bmatrix}\boldsymbol{\epsilon}_n \\ \epsilon_{n+1}\end{bmatrix} \tag{4.3}$$

$$\hat{\boldsymbol{\beta}}_{n+1} = \left({\mathbf{X}_{n+1}}^T\mathbf{X}_{n+1}\right)^{-1}{\mathbf{X}_{n+1}}^T\mathbf{y}_{n+1} \tag{4.4}$$

where $\mathbf{X}_{n+1}$ & $\mathbf{y}_{n+1}$ represent the updated input & output information containing the latest observation.

Instead of estimating $\hat{\boldsymbol{\beta}}$ using Eqn. (4.4) for every new available observation, it can be calculated recursively [40] as:

$$\mathbf{P}_{n+1} = \mathbf{P}_n - \frac{\mathbf{P}_n {\mathbf{x}_{n+1}}^T \mathbf{x}_{n+1} \mathbf{P}_n}{1 + \mathbf{x}_{n+1} \mathbf{P}_n {\mathbf{x}_{n+1}}^T} \tag{4.5}$$

$$\hat{\boldsymbol{\beta}}_{n+1} = \hat{\boldsymbol{\beta}}_n + \mathbf{P}_{n+1} {\mathbf{x}_{n+1}}^T \left(y_{n+1} - \mathbf{x}_{n+1} \hat{\boldsymbol{\beta}}_n \right) \tag{4.6}$$

where $\mathbf{P}_n = \left({\mathbf{X}_n}^T \mathbf{X}_n \right)^{-1}$. This is the recursive formulation of the linear least squares regression algorithm called RLS. It is a simple and computationally efficient technique for the cases where the regression vector, $\boldsymbol{\beta}$, is a function of time. However, as the value of n becomes larger, the influence of new observations decreases and the ability to track the changes in $\boldsymbol{\beta}$ is lost. To mitigate this, a forgetting factor λ [42], is introduced into Eqn. (4.5) as follows:

$$\mathbf{P}_{n+1} = \frac{1}{\lambda} \left(\mathbf{P}_n - \frac{\mathbf{P}_n {\mathbf{x}_{n+1}}^T \mathbf{x}_{n+1} \mathbf{P}_n}{\lambda + \mathbf{x}_{n+1} \mathbf{P}_n {\mathbf{x}_{n+1}}^T} \right) \tag{4.7}$$

where $\lambda \in [0, 1]$. The above is the RLS algorithm with a forgetting factor and is a widely used adaptive regression technique. The role of λ is to discount the past data and emphasize new samples. For small values of λ, the model will be highly adaptive to abrupt process changes but less robust to noise and outliers. Also model prediction will suffer since it means discounting past knowledge. In the extreme case, a very small value of λ will mean discounting all samples except the latest and will consequently result in a large prediction variance. On the other hand, large values of λ will give a smooth and robust but biased prediction as it will be unable to adapt quickly to process changes. Hence, λ is the bias-variance tradeoff parameter and appropriate selection of its value is critical.

4.2.2 Moving window least squares (MWLS)

The moving window formulation of the ordinary least squares algorithm is another popular adaptive parameter estimation technique. In this formulation, the model parameters are estimated again when a given number, s (step size), of new data samples have been collected. The number of samples used for model training is called the window size, w [7]. Suppose that initially we have n samples and an initial

estimate of the model parameters. Next, all but the most recent w samples are discarded. Every time a new sample becomes available, it is stored and the oldest one removed. This keeps the database size fixed at w. One can also think of this as a window of width w sliding on the stream of data, hence the name, moving window. Once s new samples have been collected, the model is re-trained on the current database.

The moving window formulation of the least squares estimate for the system described by Eqn. (4.1) is written as:

$$\begin{bmatrix} y_{n+s-w+1} \\ y_{n+s-w+2} \\ . \\ . \\ . \\ y_{n+s} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{n+s-w+1} \\ \mathbf{x}_{n+s-w+2} \\ . \\ . \\ . \\ \mathbf{x}_{n+s} \end{bmatrix} \boldsymbol{\beta}_w + \begin{bmatrix} \epsilon_{n+s-w+1} \\ \epsilon_{n+s-w+2} \\ . \\ . \\ . \\ \epsilon_{n+1} \end{bmatrix} \tag{4.8}$$

Writing Eqn. (4.8) in matrix form:

$$\hat{\boldsymbol{\beta}}_w = \left(\mathbf{X}_w{}^T \mathbf{X}_w\right)^{-1} \mathbf{X}_w{}^T \mathbf{y}_w \tag{4.9}$$

where $\mathbf{X}_w$ & $\mathbf{y}_w$ denote the latest w samples and $\hat{\boldsymbol{\beta}}_w$ denotes the least squares estimate on this window of w points. When another step has been taken (i.e., s new samples received), the regression vector is re-estimated. The length of the window signifies the size of the database which is used for parameter estimation and the step size signifies the frequency of the estimation. If s is set to 1, the model is re-trained sample-wise (i.e., as soon as a new sample is available) and for larger values of s, the model is re-trained in a blockwise fashion [7]. Choosing both the parameters, w and s, is critical as inappropriate setting can lead to decrease in performance [43]. Together, they play a role similar to the role of λ in RLS. Large window size means less adaptability, more robustness and more computation whereas the opposite is true of a small window size. Selecting a large step size leads to lower computation cost since it means decreased frequency of model training but it also means that the model will be slow to respond to system changes. A model with a low s value will be quicker to respond to changes but the high frequency of model training will cause the computation cost to rise.

4.2.3 Online Passive Aggressive Algorithm (OPAA)

Here, we introduce the Online Passive Aggressive Algorithm (OPAA) for regression problems [38], an adaptive parameter tracking algorithm from the machine learning literature. Similar to the scenario for the formulation of the RLS algorithm, suppose we have n observations and an estimate for the regression vector in Eqn. (4.1) as $\hat{\boldsymbol{\beta}}_n$. The recursive update to $\hat{\boldsymbol{\beta}}_n$ is then obtained as follows:

- Calculate the loss, l_{n+1}, based on the error of the current prediction from the following loss function [38]:

$$l_{\xi}(\hat{\boldsymbol{\beta}}_n) = \begin{cases} 0 & |y_{n+1} - \mathbf{x}_{n+1}\hat{\boldsymbol{\beta}}_n| \leq \xi \\ |y_{n+1} - \mathbf{x}_{n+1}\hat{\boldsymbol{\beta}}_n| - \xi & \text{otherwise} \end{cases} \tag{4.10}$$

  where $\mathbf{x}_{n+1}\hat{\boldsymbol{\beta}}_n$ gives the prediction, $\hat{y}_{n+1}$, corresponding to the latest predictor variable $\mathbf{x}_{n+1}$. This loss is zero when the predicted target deviates from the true target by less than ξ and otherwise grows linearly with $|y_{n+1} - \hat{y}_{n+1}|$. The threshold parameter, ξ , is a positive real number that controls the sensitivity of the algorithm to inaccuracies in the prediction.

- Next, find the new updated regression vector, $\hat{\boldsymbol{\beta}}_{n+1}$ such that the loss for the current term is zero while minimizing the distance of $\hat{\boldsymbol{\beta}}_{n+1}$ from $\hat{\boldsymbol{\beta}}_n$ [38].

  Hence, the update to $\hat{\boldsymbol{\beta}}_n$ is made as follows:

$$\hat{\boldsymbol{\beta}}_{n+1} = \arg\min_{\boldsymbol{\beta}} \frac{||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n||^2}{2} \quad s.t. \quad l_{\xi}(\boldsymbol{\beta}) = 0 \tag{4.11}$$

  Using Lagrangian optimization, the closed form expression for the updated regression vector is obtained as:

$$\hat{\boldsymbol{\beta}}_{n+1} = \hat{\boldsymbol{\beta}}_n + \text{sign}(y_{n+1} - \hat{y}_{n+1})\,\mathbf{x}_{n+1}\tau_{n+1} \tag{4.12}$$

  where $\tau_{n+1} = \frac{l_{\xi}(\hat{\boldsymbol{\beta}}_n)}{||\mathbf{x}_{n+1}||^2}$

From Eqn. (4.12) we see that the change in $\hat{\boldsymbol{\beta}}_n$ is proportional to τ_{n+1}. Crammer et al. [38] also introduce two variants of the update strategy called OPAA-I & OPAA-II respectively. The only difference is in the computation of τ_{n+1}, which for the two cases respectively is:

$$\text{OPAA-I: } \tau_{n+1} = min\{C, \frac{l_\xi(\hat{\boldsymbol{\beta}}_n)}{||\mathbf{x}_{n+1}||^2}\} \tag{4.13}$$

$$\text{OPAA-II: } \tau_{n+1} = \frac{l_\xi(\hat{\boldsymbol{\beta}}_n)}{||\mathbf{x}_{n+1}||^2 + \frac{1}{2C}} \tag{4.14}$$

where C is a positive parameter that controls the aggressiveness of the update to $\hat{\boldsymbol{\beta}}_n$. For very large values of C, OPAA-I & OPAA-II reduce to the original OPAA algorithm while smaller C values cause a less aggressive update.

4.3 Proposed algorithm

Before proceeding further, following remarks about the parameter update strategies used in RLS/MWLS and OPAA are in order:

1. The form of the update to $\hat{\boldsymbol{\beta}}$ in both RLS and OPAA is very similar. In both cases, the magnitude of the change to $\hat{\boldsymbol{\beta}}$ is proportional to the error in prediction using the previous estimate of $\hat{\boldsymbol{\beta}}$. However, one difference is readily observed. Whereas in RLS, the update occurs for each and every prediction error and for every step size in MWLS, in the case of OPAA, the update occurs only when the prediction deviates from the target by more than the threshold, ξ. This can be an advantage in situations where it is undesirable for minor process fluctuations to cause changes in the parameter estimates.

2. In the case of the RLS, the update term takes into account the history of the input variables in the form of the covariance matrix. This does not happen in the case of the OPAA and the update considers only the prediction error for the current input variable.

3. The RLS algorithm minimizes the squared error cost function for all the samples in the database and the MWLS does the same for the samples in the window.

The OPAA, on the other hand, minimizes within a threshold, the absolute value of the prediction error for the latest sample. Thus, the RLS algorithm will be sensitive to large errors because of the squared error cost function. The OPAA, due to the absolute deviation loss term and cautious update will tend to be robust. However, since it updates $\hat{\boldsymbol{\beta}}$ based on the performance of a single sample, it is still vulnerable to arbitrary process fluctuations.

From the above observations we see that OPAA has certain attractive features as a predictive algorithm. However, in its current form it is not ideally suited for use in practical soft sensor design. We propose certain modifications to OPAA and call it SPAA (smoothed passive aggressive algorithm) for reasons that will be evident later.

4.3.1 Smoothed Passive Aggressive Algorithm (SPAA)

It was highlighted previously that OPAA uses a single observation for its parameter update. It is a generally accepted rule that overfitting occurs when the ratio of model parameters to training data size is very high. Since, in OPAA, the regression parameter has to always fit the latest observation, it is equivalent to using a single sample for model training. It has also been pointed by Kim et al. [14], that overfitting occurs when the bandwidth of the local model used in Just-In-Time models is very low, i.e., very few points participate in building the local model. For industrial applications therefore, it is generally required to have a minimum sample size for model training to avoid overfitting and high prediction variance. We deal with this issue by modifying the loss function in OPAA to take account of a window of points instead of a single sample. The new formulation based on a window will be reduced to the original formulation if w is set as 1. The new loss term is formulated as follows:

$$l_e\left(\hat{\boldsymbol{\beta}}_n\right) = \begin{cases} 0 & \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i\hat{\boldsymbol{\beta}}_n| \leq sad_w + e \\ \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i\hat{\boldsymbol{\beta}}_n| - (sad_w + e) & \text{otherwise} \end{cases} \tag{4.15}$$

where, sad_w is the sum of the absolute deviations given by the least absolute deviations (LAD) solution for the set of w points contained in the window, $\{(\mathbf{x}_{n-w+2}, y_{n-w+2}), \ldots (\mathbf{x}_{n+1}, y_{n+1})\}$, i.e.

$$sad_w = \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i \boldsymbol{\beta}_{lad}| \tag{4.16}$$

$$\boldsymbol{\beta}_{lad} = \arg\min_{\boldsymbol{\beta}} \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i \boldsymbol{\beta}_{lad}| \tag{4.17}$$

e, the threshold parameter for SPAA, is a % (percentage) of the sad_w value for the current window. The reader is reminded that the threshold ξ in OPAA described earlier is not calculated as a percentage, though both, ξ and e, serve a similar purpose. Finally, the update to $\hat{\boldsymbol{\beta}}$, the regression vector, is obtained as follows:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{n+1} &= \arg\min_{\boldsymbol{\beta}} \frac{||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n||^2}{2} \quad s.t. \quad l_e(\boldsymbol{\beta}, \mathbf{X}_w, \mathbf{y}_w) = 0 \quad or \\ \hat{\boldsymbol{\beta}}_{n+1} &= \arg\min_{\boldsymbol{\beta}} \frac{||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n||^2}{2} \; s.t. \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i \boldsymbol{\beta}| \leq sad_w + e \end{aligned} \tag{4.18}$$

Before analyzing SPAA, the solution strategy for Eqn. (4.18) is discussed. The loss function in SPAA requires the LAD solution for the current window, which can be calculated by converting the optimization to a linear program [44, 45] as:

$$\begin{aligned} \boldsymbol{\beta}_{lad} &= \arg\min_{\boldsymbol{\beta}} \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i \boldsymbol{\beta}| \quad or \\ \boldsymbol{\beta}_{lad} &= \arg\min_{\boldsymbol{\beta}, u_i} \sum_{i=n-w+2}^{n+1} u_i \qquad \text{s.t.} \qquad u_i - |y_i - \mathbf{x}_i \boldsymbol{\beta}| = 0 \end{aligned} \tag{4.19}$$

Finally, the above can be written as the following linear program:

$$\begin{aligned} \boldsymbol{\beta}_{lad} &= \arg\min_{\boldsymbol{\beta}, u_i} \sum_{i=n-w+2}^{n+1} u_i \\ u_i &\geq (y_i - \mathbf{x}_i \boldsymbol{\beta}), \quad u_i \geq -(y_i - \mathbf{x}_i \boldsymbol{\beta}) \end{aligned} \tag{4.20}$$

where, u_i are artificial variables introduced to convert the problem to a linear program. The inequality in Eqn. (4.20) forces each u_i to equal $|y_i - \mathbf{x}_i \boldsymbol{\beta}|$ upon minimization. Once, $\boldsymbol{\beta}_{lad}$ is found, sad_w required in evaluating the loss function in SPAA can be obtained. Similar to Eqn. (4.20), the overall optimization in Eqn. (4.18) can be

reduced to a set of linear inequality constraints with a quadratic objective function as:

$$\begin{aligned} &\hat{\boldsymbol{\beta}}_{n+1} = \arg\min_{\boldsymbol{\beta}} \frac{||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n||^2}{2} \quad \text{subject to the constraints :} \\ &u_i \geq (y_i - \mathbf{x}_i\boldsymbol{\beta}), u_i \geq -(y_i - \mathbf{x}_i\boldsymbol{\beta}), \sum_{i=n-w+2}^{n+1} u_i \leq sad_w + e \end{aligned} \tag{4.21}$$

The variables for this optimization are the u_i's and $\boldsymbol{\beta}$. Since we already know that there exists a $\boldsymbol{\beta} = \boldsymbol{\beta}_{lad}$ that leads to $\sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i\boldsymbol{\beta}_{lad}| = sad_w$, it is guaranteed that we can always find a $\boldsymbol{\beta}$ which will satisfy the inequalities in Eqn. (4.21) above. As $e \geq 0$, we have:

$$sad_w \;\leq\; \sum_i |y_i - \mathbf{x}_i\boldsymbol{\beta}| \;\leq\; \sum_i u_i \;\leq\; sad_w + e$$

However, unlike in Eqn. (4.20), Eqn. (4.21) does not always force each u_i to equal $|y_i - \mathbf{x}_i\boldsymbol{\beta}|$, but the original requirements as per Eqn. (4.18) are always met. The inequality constraints ensure that the new total deviation, $\sum_i |y_i - \mathbf{x}_i|$, is within the acceptable relaxed limit of $(sad_w + e)$. The use of the threshold parameter, e, causes existence of multiple solutions for $\hat{\boldsymbol{\beta}}$ and u_i's satisfying the constraints. The objective function then causes the selection of that $\hat{\boldsymbol{\beta}}$ which is closest to $\hat{\boldsymbol{\beta}}_n$, hence giving a unique solution. Were one to set e to zero, $\hat{\boldsymbol{\beta}}$ equals $\boldsymbol{\beta}_{lad}$. Depending on whether $\boldsymbol{\beta}_{lad}$ is unique or not (there are situations where $\boldsymbol{\beta}_{lad}$ may be non-unique), the objective function then becomes inactive or active. Hence the formulation in Eqn. (4.18), among other things, always ensures a unique update.

We see that, given some initial estimate, $\hat{\boldsymbol{\beta}}$ is learnt adaptively from the data. There are two parameters, w & e, the window size and the threshold respectively that are needed for the update equation. These can be learnt offline by minimizing the prediction error on the training data set. The significance of these parameters will be discussed in the following section.

4.3.2 Analysis, comparison and comments

Let us now perform a comparative analysis of the proposed SPAA algorithm with the original form and with the RLS and MWLS algorithms. It is also interesting to note that SPAA with a window size of 1 reduces to OPAA, and with e as 0% (meaning aggressive parameter update), reduces to a moving window LAD algorithm. Therefore, SPAA can also be called a moving window L_1 or LAD algorithm with a cautious parameter update strategy or alternately a window based OPAA algorithm.

SPAA and OPAA

The OPAA algorithm, as shown in Eqn. (4.11), contains two terms in the update expression for the regression vector, the left hand/objective term and the right hand/constraint term. The constraint term defines a feasible space for $\hat{\boldsymbol{\beta}}$. This space contains $\hat{\boldsymbol{\beta}}$ values, all of which satisfy the required performance in prediction accuracy. In OPAA, this requirement in prediction accuracy is measured by the ability of $\hat{\boldsymbol{\beta}}$ to correctly predict the latest response variable, i.e. y_{n+1}. From this space of possible $\hat{\boldsymbol{\beta}}$ values, the one closest to its previous value is picked. The sensitivity/threshold parameter, ξ, can be considered as a relaxation to the requirement in prediction accuracy and its value will depend on the particular application. For example, when an appropriate value of ξ is selected, noise or minor process fluctuations will not cause any undesirable changes in the model parameters. This happens because $\hat{\boldsymbol{\beta}}_n$ correctly predicts y_{n+1} within the desired threshold ξ, and forms a part of the feasible space for $\hat{\boldsymbol{\beta}}$. The objective function is then minimized for the case when $\hat{\boldsymbol{\beta}}_n$ remains unchanged, i.e. $\hat{\boldsymbol{\beta}}_{n+1} = \hat{\boldsymbol{\beta}}_n$ and no update occurs. Only when large changes in the system occur, will the decrease in the accuracy of $\hat{\boldsymbol{\beta}}_n$ be large enough to cause a change in its value. The optimum value of ξ can be learnt by minimizing the error on the training data. The key point to note here is that the model update is based only on the most recent sample. This is true in the case of the two variants, OPAA-I & OPAA-II as well. Hence, a single abnormal data sample can have a large impact on the performance.

In contrast to OPAA, the prediction performance over the latest w observations is considered for parameter updates in SPAA. Generally in practical applications, model training is always carried out over a minimum number of samples to avoid issues like

overfitting and high prediction variance. From Eqn. (4.18), we can see that this performance is measured by the sum of the absolute deviations in predicting the points in a window. We can consider OPAA as a specific case of SPAA with w as 1. The use of a window instead of a single sample to assess the performance of the model makes it more robust to arbitrary fluctuations in the system. It acts in a way similar to the forgetting factor used in RLS and its selection is a tradeoff between prediction bias and prediction variance or model robustness and model adaptability. Models with a larger window size will be more robust and slower to adapt whereas those with a smaller window size will be less robust and faster to adapt. Hence, SPAA with an appropriately selected window size performs better than OPAA or either of its variants, OPAA-I or OPAA-II. This will be demonstrated in the applications section later on. The down-side of using a window formulation is that the closed form update expression of the original algorithm is lost and computational complexity is increased. Although using a window will lead to robustness, SPAA without the sensitivity parameter, e, will still be subjected to updates caused by noise or minor disturbances. This is evident from Eqn. (4.20), since the sad_w for every window will almost always be different from each other. The use of e, as of ξ in OPAA, causes the loss term to become zero for minor fluctuations and avoids unnecessary updates. Again, offline optimization on the training data is one way to select w & e values. Expert process knowledge may be used to restrict w within a meaningful range. For values of e larger than the optimal one, the loss term will be zero more frequently and hence will lead to fewer updates which means lower computation since the optimization in Eqn. (4.18) will be no longer required. This will be reflected in a decreased performance in terms of R and rmsep values. However, because of the window formulation it would still be able to capture a trend change, though with a slight delay and bias depending on e.

Hence, the SPAA formulation gives a general framework where the w and e values can be tweaked as per the application at hand. For example, in situations where capturing the overall trend with increased robustness is preferred rather than exactly predicting the reference values (possibly laboratory values in the case of soft sensor applications), an e value greater than the optimum one (optimum in the sense of minimum training error) could be chosen. This tradeoff between computation, robustness

and accuracy can be suited to match the requirements.

SPAA and RLS/MWLS

As will be shown in the next section, the LAD estimate is the ML (maximum likelihood) estimate when the residuals are assumed to follow a Laplace distribution. The least squares estimate on the other hand is based on the residuals following a Gaussian distribution. Hence, the LAD estimate, and consequently the SPAA algorithm based on it, is more robust to outlying values in the output than the least squares algorithms [46]. Furthermore, the regression vector is updated for every new sample in the case of RLS and for every update step in the case of MWLS. The SPAA on the other hand can be said to follow a smart update since it updates the regression parameters only when the average performance over the latest w points is unsatisfactory. Also, since it uses the absolute deviation loss function and does a cautious update, it is to be expected that the SPAA algorithm will have a fewer number of predictions with high relative error values compared to the other algorithms. On the computation side however, based on computer run time during simulations, SPAA is computationally heavier than either RLS or MWLS.

SPAA and LAD

SPAA can also be seen as a modified implementation of the LAD algorithm. The LAD estimate is the ML estimate when the residuals are assumed to follow a Laplace (or double exponential) distribution [36, 47]. Consider a linear system similar to Eqn. (4.1) where the errors are independent and identically distributed according to the zero mean Laplace distribution:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim L\left(0, b\right) \tag{4.22}$$

The density function of the errors is given as:

$$\epsilon_i = \frac{1}{2b} \exp\left(-\frac{|\epsilon_i|}{b}\right) \tag{4.23}$$

where b is the scale parameter.

Writing the likelihood function we have:

$$l_{\boldsymbol{\beta},b} = f\left(\mathbf{y}, \mathbf{X}|\boldsymbol{\beta}, b\right) = f\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, b\right) f\left(\mathbf{X}|\boldsymbol{\beta}, b\right) \tag{4.24}$$

Considering that the predictor variables and the regression parameters are independent of each other, the last term in the above equation is a constant and can be denoted as c.

$$l_{\boldsymbol{\beta},b} = cf\left(y_1, y_2, ...y_n|\mathbf{X}, \boldsymbol{\beta}, b\right) = \frac{c}{(2b)^n} \exp\left(-\frac{\sum |y_i - \mathbf{x}_i\boldsymbol{\beta}|}{b}\right) \tag{4.25}$$

Finally, the log-likelihood function is given as:

$$L_{\boldsymbol{\beta},b} = \ln l_{\boldsymbol{\beta},b} = -\frac{\sum |y_i - \mathbf{x}_i\boldsymbol{\beta}|}{b} \; + \; \ln c \; - \; n \ln 2b \tag{4.26}$$

Thus, maximizing the likelihood is equivalent to minimizing $\sum |y_i - \mathbf{x}_i\boldsymbol{\beta}|$, which is the LAD solution. Since the Laplace distribution has fatter tails compared to the Gaussian distribution, the LAD solution is more robust to outliers in the output compared to the least squares solution. However, the LAD method also has certain characteristics that make it unsuitable for implementation, the primary of which are the non-uniqueness of the solution and the computational complexity compared to least squares [48, 49]. In the case of the least squares solution, the condition required for uniqueness is for the input matrix to have full column rank. However, there is no particular defined condition that can guarantee a unique LAD solution [50]. Secondly, the simplex method, which is typically used to solve linear programs, becomes slow for large observations. Nevertheless, upto a few hundred observations, LAD regression is competitive with LS [51].

The proposed algorithm, SPAA, is able to retain the desirable characteristics of the LAD solution while tackling the above mentioned disadvantages. Although the size of the window, w , will vary with the application, it will rarely be large enough for computation to become a factor in its implementation. Secondly, the form of the optimization in Eqn. (4.18) inherently deals with the issue of multiple solutions by selecting the regression vector with the least distance from its previous value.

4.3.3 SPAA variants

The SPAA framework is quite flexible and a number of variations are possible within it. These variations come at almost no increase in computational complexity. When to use a particular variant depends on the application and availability of expert process knowledge. As mentioned earlier, like OPAA, the SPAA formulation consists of two components, the objective term which is to be minimized and the constraint term. The variations in these two terms are now discussed respectively.

Objective function variations

There are numerous ways to describe closeness in mathematical terms besides the one used for $\hat{\boldsymbol{\beta}}$ in OPAA. We discuss some alternatives below.

(a) Sum of Squares of Fractional Change
Instead of minimizing $\frac{||\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_n||^2}{2}$ in the objective, one could minimize, $\sum_j^m \frac{1}{2}\left(\frac{\beta_j-\hat{\beta}_{n,j}}{\hat{\beta}_{n,j}}\right)^2$, i.e. the sum of squares of the fractional changes in the regression parameters. If the update in $\hat{\boldsymbol{\beta}}$ is brought about by a disturbance or outlier, the former should perform better whereas for a system change the latter should be better. This is because in the case of a disturbance/outlier, one would expect the parameter to return to normal and the error will be minimized when the change in the regression coefficients (and not the fractional change) due to the disturbance is minimal. On the other hand, a system change implies a new value of $\hat{\boldsymbol{\beta}}$. In this case, among all $\hat{\boldsymbol{\beta}}$ that satisfy the constraint, selecting the one that minimizes the relative change in regression coefficients makes more sense. In applications where regression coefficients indicate a physical relationship between response and explanatory variables, it is more reasonable to accept, say, a 5 % change in a coefficient with a large magnitude than accept a 100 % change in one with a much smaller magnitude. Selecting between the two depends on the expected noise/disturbance and the trend changes anticipated in the system and use of expert/prior knowledge, if available, is recommended.

(b) L^1 norm
Here, instead of minimizing the L^2 norm in the objective, one could minimize the L^1

norm. Hence, the update to $\hat{\boldsymbol{\beta}}$ now becomes:

$$\hat{\boldsymbol{\beta}}_{n+1} = \arg\min_{\boldsymbol{\beta}} |\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n|_1 \quad s.t. \quad \sum_{i=n-w+2}^{n+1} |y_i - \mathbf{x}_i\boldsymbol{\beta}| \leq sad_w + e \tag{4.27}$$

$|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n|_1$ is the L^1 norm of the vector $(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n)$ and is the same as $\sum_{j=1}^{m} |\beta_j - \hat{\beta}_{n,j}|$, where m is the dimension of $\hat{\boldsymbol{\beta}}$. Without going into details, we directly write the form of the final update expression. The reader can verify this using the techniques employed previously in Section 4.3.1.

$$\begin{aligned}
&\hat{\boldsymbol{\beta}}_{n+1} = \arg\min_{\boldsymbol{\beta}, z_j, u_i} \sum_{j=1}^{m} z_j \quad \text{s.t.} \\
&z_j \geq \left(\beta_j - \hat{\beta}_{n,j}\right), \quad z_j \geq -\left(\beta_j - \hat{\beta}_{n,j}\right) \quad \& \\
&u_i \geq \left(y_i - \mathbf{x}_i\boldsymbol{\beta}\right), u_i \geq -\left(y_i - \mathbf{x}_i\boldsymbol{\beta}\right), \sum_{i=n-w+2}^{n+1} u_i \leq sad_w + e
\end{aligned} \tag{4.28}$$

where, z_i & u_i are artificial variables introduced for conversion into a linear program. The consequence of using the L^1 norm instead of the L^2 norm is that because of the sparseness property of the L^1 norm, the update with changes in fewer of the regression coefficients will be preferred whereas in the L^2 norm, the update will almost always be one where all the coefficients change to some extent. Also, since the prediction is a linear function of $\hat{\boldsymbol{\beta}}$, using the L^1 norm will cause less deviation from the previous $\hat{\boldsymbol{\beta}}$ value. Again, this may or may not be desirable depending on whether the update is caused by a disturbance or process change and use of expert judgment will be required to select between the two.

Constraint term variations

Lastly, as pointed out by Fisher [52], the use of LAD for regression is very flexible. Since the SPAA algorithm, in its constraint/loss term, uses a slightly modified version of the absolute cost function, it also inherits the same flexibility. A number of constraints can be easily incorporated within the existing optimization problem without any significant increase in computational requirement. For example, if it is

required to weigh the data unequally, it can be done by simply transforming the data as, $\left(\widetilde{\mathbf{x}_i}, \widetilde{y_i}\right) = w_i\left(\mathbf{x}_i, y_i\right)$ [53]. The weights, w_i could be time weights [54], or they could be a robust measure of the distances in the input space [53].

Also, in the case of industrial applications, use of prior or expert knowledge regarding the systems can be implemented easily. The regression coefficients in the case of real world systems quantify the physical relationship between the response and explanatory variables. Hence, suppose that based on experience or historical evidence from similar scenarios elsewhere, one wanted to restrict the fractional change in any particular regression coefficient, j, to a value p , the same can be very easily incorporated into the existing framework by adding the additional inequality constraint as follows:

$$\beta_j \leq (1+p)\,\hat{\beta}_{n,j} \quad \& \quad \beta_j \geq (1-p)\,\hat{\beta}_{n,j} \tag{4.29}$$

Another variation in the constraint term can be made by employing a quadratic loss function. Keeping the objective function unchanged, the new update expression can be written as:

$$\hat{\boldsymbol{\beta}}_{n+1} = \arg\min_{\boldsymbol{\beta}} \frac{\left|\left|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_n\right|\right|^2}{2} \qquad s.t. \qquad \sum_{i=n-w+2}^{n+1} \left(y_i - \mathbf{x}_i\boldsymbol{\beta}\right)^2 \leq ls_w + e \tag{4.30}$$

Here, ls_w is the sum of squares of deviations given by the least squares solution on the window and e is the threshold/sensitivity parameter. Using the method of Lagrangian multiplier, the above leads to a set of quadratic equations for the update expression as opposed to the original SPAA which does not have an analytical update expression. However, with the decrease in computational complexity, the robustness of the absolute value cost function is also lost.

4.4 Results and discussion

In this section, the SPAA algorithm is tested on a numerically simulated and an industrial data set and its performance compared with OPAA, RLS and MWLS. The

results are consistent with the observations and comments made earlier and clearly bring out the advantages of SPAA.

4.4.1 Numerical simulation

A SISO (single input single output) parameter varying ARX model is simulated as follows:

$$y_t = a_0 + a_1 y_{t-1} + a_2 u_t + a_3 u_{t-1} + \epsilon_t \tag{4.31}$$

where u_t, the input, is a uniformly distributed random number in the interval $[-2, +2]$, $a_0,\ a_1,\ a_2$ & a_3 are the ARX model parameters & ϵ_t is zero mean white noise. For the training set, a total of 600 samples are generated and the ARX parameters varied with a combination of ramp and step changes as shown in Table 4.1 (all ramp changes are linear).

Table 4.1: Parameter variation: Numerical simulation - Training set

time, t	a_0	a_1	a_2	a_3
001 : 200	-1.00 $\rightarrow$ 0.00	+0.50	+1.00 $\rightarrow$ +0.25	+0.25
201 : 400	0.00	-0.50	+0.25	+1.00
401 : 600	0.00 $\rightarrow$ +1.00	+0.75	+0.25 $\rightarrow$ -0.75	-0.50

All algorithms are trained on this data set and the optimum parameters (w, ξ, C, e, λ) selected by minimizing the rmsep on it . The least squares solution for the first 20 points is used as the initial value of the ARX parameters. Next, for the test set, two cases are considered.

(i) Case I:

The same equation, Eqn. (4.31) used for the training set is used to generate the test set.

(ii) Case II

The sequence in which the ARX model parameters vary in the training set is changed for the test set generation in this case. The parameter variation, shown in Table 4.2, is the same as in Table 4.1 but with rows 1 and 2 interchanged.

In both test cases, outliers are added to the output, y. It is then observed how the different algorithms perform in the presence of extreme values. For the test cases, the

Table 4.2: Parameter variation: Numerical simulation - Test set - Case II

time, t	a_0	a_1	a_2	a_3
001 : 200	0.00	-0.50	+0.25	+1.00
201 : 400	0.00 $\rightarrow$ -1.00	+0.50	+0.25 $\rightarrow$ +1.00	+0.25
401 : 600	-1.00 $\rightarrow$ +1.00	+0.75	+1.00 $\rightarrow$ -0.75	-0.50

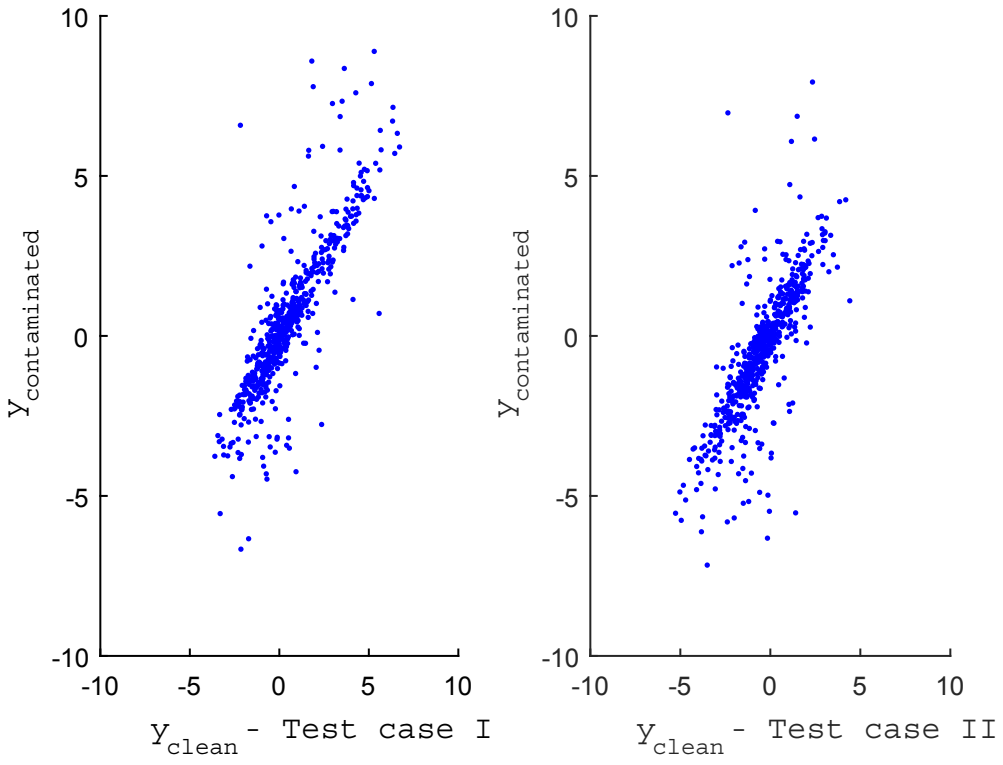

Figure 4.1: Contaminated vs. clean output for Test Cases I and II

results are given based on comparison of the predicted output with the outlier free output. Figure 4.1 presents the scatter plot of the outputs with outliers against the clean data.

Before the results are presented, we show Figures, 4.2 & 4.3 illustrating the effects of the parameters e and w. From Figures 4.2 & 4.3, one can see that w and e affect parameter estimation, and hence prediction, differently. While both have a smoothing effect on the estimation and prediction, increase in w causes increasing influence of past samples on the current estimation. Thus, a large window leads to a robust but less adaptive estimation. Once a trend change is detected, it leads to a bias in the estimation. However, this bias is removed when the sliding window moves forward

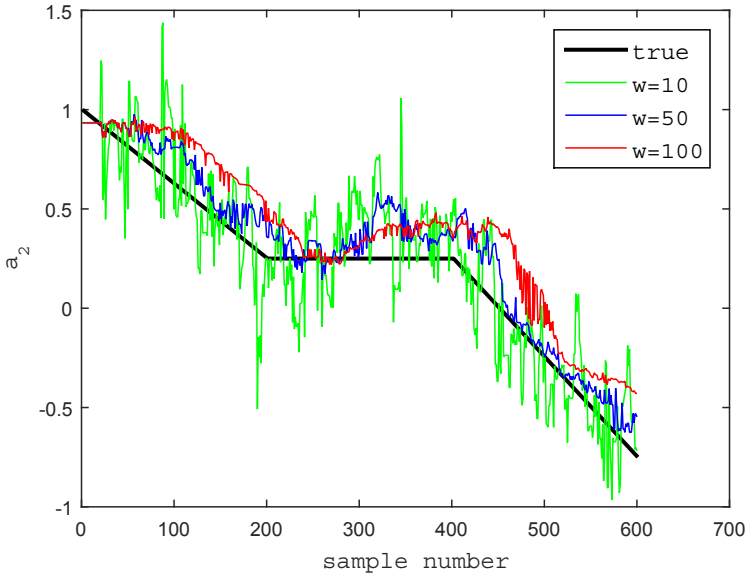

Figure 4.2: Effect of w on estimating a_2, for fixed $e(= 0)$

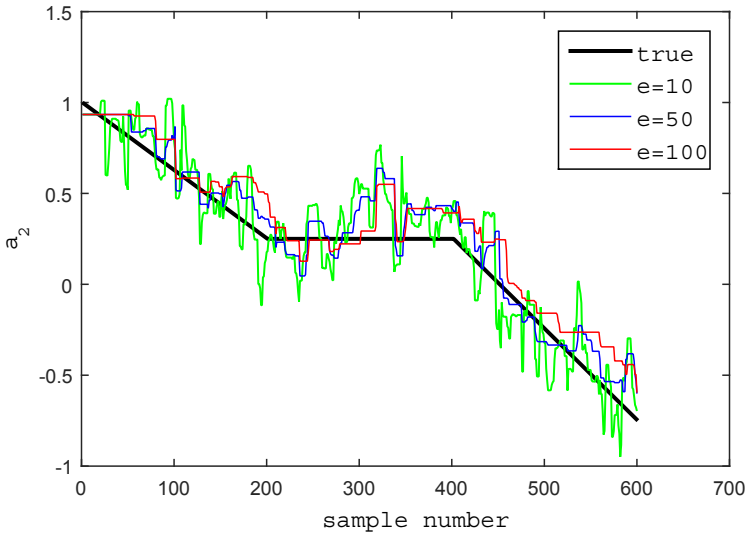

Figure 4.3: Effect of e on estimating a_2, for fixed $w(= 10)$

and leaves all past samples behind. This role is quite similar to that of λ in RLS. On the other hand, increase in e results in less parameter updates but it also causes a delay in detecting the trend change. As soon as this trend change is detected, (by activation of the loss function), the estimation of the updated parameters is based only on the current window. The bias is caused by the presence of the threshold since according to the formulation, the updated parameter does not have to fit the LAD solution for the current window exactly.

Table 4.3: Test Results: Numerical simulation - Case I

Model	$\lambda/\xi/e/w$	R	rmsep (y)	Updates	rmsep($a_0/a_1/a_2/a_3$)
RLS	0.94/-/-/-	0.941	0.695	580	0.38/0.33/0.19/0.30
MWLS	-/-/-/30	0.935	0.731	580	0.44/0.33/0.22/0.32
SPAA	-/-/20/20	0.947	0.648	125	0.28/0.30/0.16/0.29
OPAA	-/0.8/-/-	0.845	1.223	301	0.51/0.42/0.54/0.50

Table 4.4: Test Results: Numerical simulation - Case II

Model	$\lambda/\xi/e/w$	R	rmsep (y)	Updates	rmsep($a_0/a_1/a_2/a_3$)
RLS	0.94/-/-/-	0.903	0.757	580	0.34/0.38/0.28/0.29
MWLS	-/-/-/30	0.892	0.802	580	0.51/0.44/0.29/0.35
SPAA	-/-/20/20	0.926	0.657	126	0.29/0.27/0.21/0.21
OPAA	-/0.8/-/-	0.752	1.375	306	0.95/0.77/0.53/0.54

Tables 4.3 & 4.4 show the performance of the various algorithms for Test Cases I & II. As discussed earlier, SPAA with $e = 0$ is effectively a moving window LAD algorithm. The rmsep values of SPAA ($e = 0$) for Test Cases I & II are 0.658 & 0.697 respectively, which are higher than those of SPAA ($e = 20$) for the corresponding cases. For the variants of OPAA, rmsep values of 0.912/0.944 for OPAA-I, & 1.076/1.166 for OPAA-II are obtained for Test Cases I/II. Although the OPAA variants show improvement, it is still much less accurate than the other algorithms because of the update being based on a single sample as pointed out earlier. Another observation is that more than one combination of C and ξ can lead to the least rmsep on the training set. In such cases it is difficult to select the appropriate set of parameters for OPAA -I/II.

The results clearly show that SPAA performs the best among all the algorithms. The greater the number/degree of outlying values, the greater will be the difference

in the performance of SPAA and the other algorithms. It is also interesting to note that the number of updates in the SPAA algorithm for both cases is significantly less than the rest, including OPAA-I and OPAA-II. Figures 4.4, 4.5, 4.6 and 4.7 show the tracking performance of RLS, MWLS, SPAA, OPAA & its variants for parameter a_0 for both test cases. The tracking performance for parameters a_1, a_2 & a_3 is also similar and the associated figures are presented in Appendix A. These figures are able to demonstrate clearly why the SPAA algorithm performs as it does. For clarity, the comparison of SPAA with RLS, MWLS and with OPAA, OPAA-I is shown separately (performance of OPAA-II is quite similar to OPAA-I and is not included in the figures).

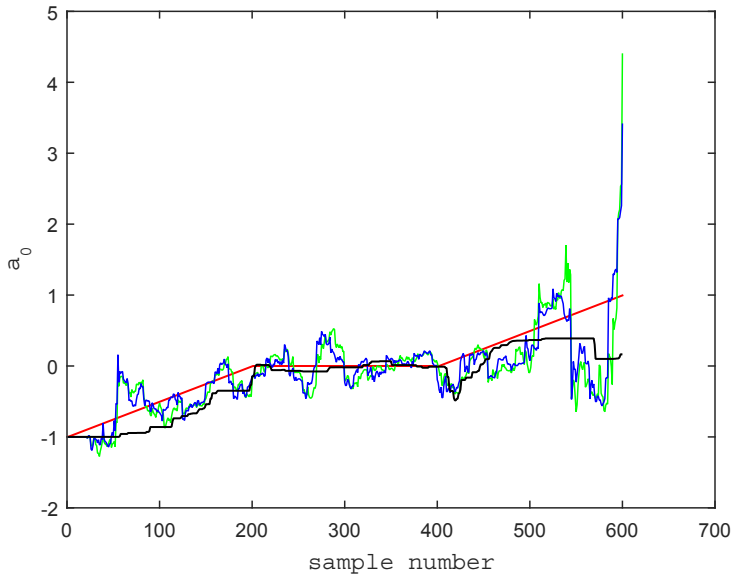

Figure 4.4: Tracking a_0, Test Case I. - True, - RLS, - MWLS, - SPAA

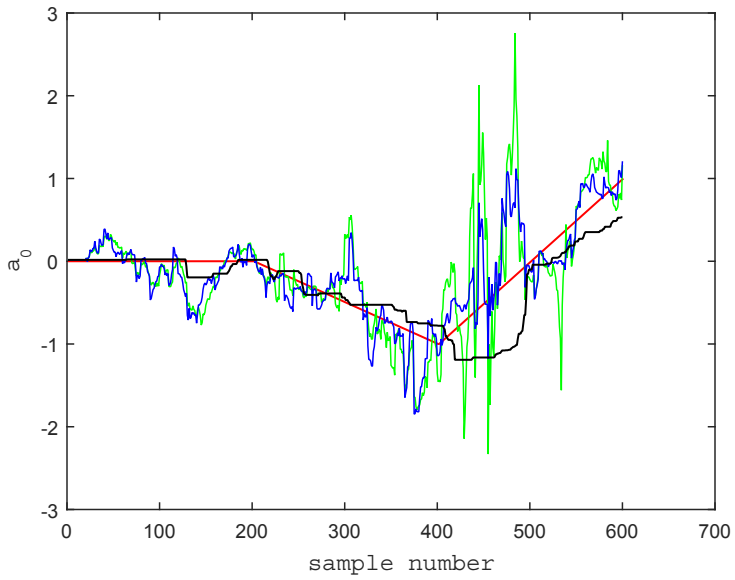

Figure 4.5: Tracking a_0, Test Case II. - True, - RLS, - MWLS, - SPAA

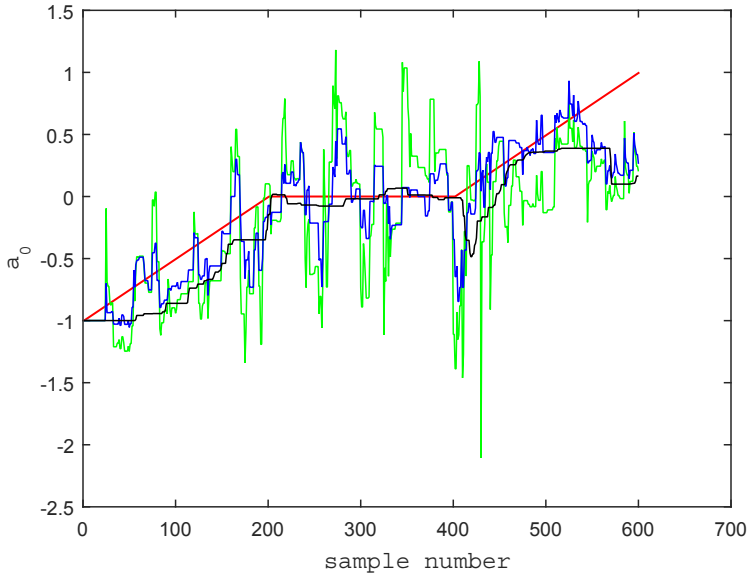

Figure 4.6: Tracking a_0, Test Case I. - True, - OPAA-I, - OPAA, - SPAA

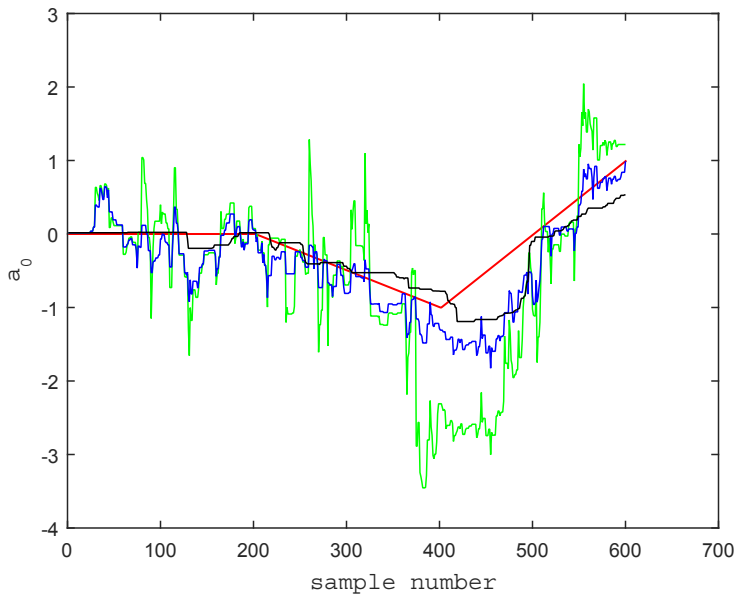

Figure 4.7: Tracking a_0, Test Case II. - True, - OPAA-I, - OPAA, - SPAA

It is clear that SPAA is much more robust than the other algorithms. However, SPAA also leads to a bias in the parameter estimation. This is possibly the reason why the difference in performance is not more in terms of the R or rmsep values. However, for industrial applications it is generally not necessary to track the true (or laboratory) values exactly (due to potential errors in laboratory analysis), merely tracking the trend of the output is satisfactory. It is also evident that compared to the other algorithms the variance of the parameter estimates in SPAA is quite low. In this regard, SPAA's advantage is brought out more clearly.

4.4.2 Industrial case study

In this section, an industrial data set from an oil sands processing plant located in Alberta, Canada is used to assess the performance of SPAA. The output is the Reid Vapor Pressure (RVP) of the bottoms of a de-propanizer column which is part of the upgrading unit of the oil sands processing plant. The inputs used are the flowrate, temperature and pressure associated with the de-propanizer column. Due to proprietary reasons, further details are not given regarding the process and normalized values are used for the inputs and output. The system is approximated by the following linear model:

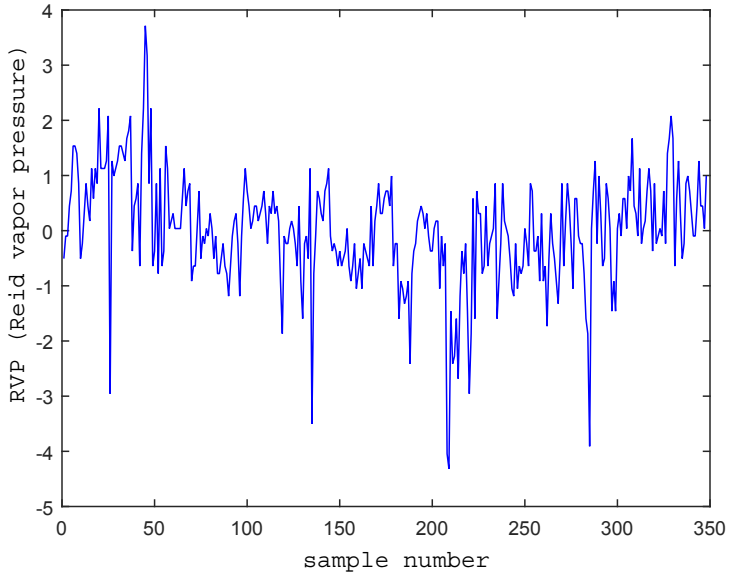

Figure 4.8: RVP normalized values,test set

$$\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\beta}} \tag{4.32}$$

Around 700 and 350 samples form the training and test sets respectively. The least squares solution based on the first 50 points is used as the initial estimate for $\boldsymbol{\beta}$. Figure 4.8 shows the plot of the normalized RVP values for the test set. The test set contains potential outliers based on the 3 σ edit rule [30] since the explanatory variables for the corresponding points in the test set are within the normal operating range.

Table 4.5 shows the performance of the algorithms on the test set. While calculating the R and rmsep values, the potential outliers were not included in the calculation in order to check the performance with respect to the outlier free data.

Table 4.5: Test Results: Industrial case study

Model	λ	e	w	R	rmsep	Updates
RLS	0.95	-	-	0.597	0.414	348
MWLS	-	-	30	0.571	0.432	348
SPAA	-	1.5	30	0.622	0.401	207

SPAA performs better than RLS and MWLS in the test set in the presence of potential outliers. A selected section of the prediction trend is displayed in Figure

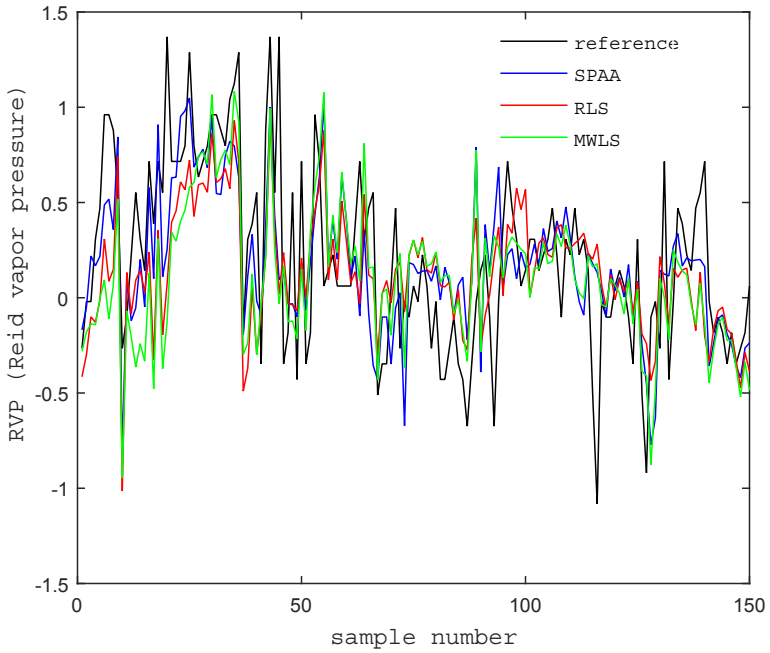

Figure 4.9: Prediction comparison on a section of test data

4.9. It shows that SPAA is able to track the reference value better in comparison to the other methods. The number of updates to $\boldsymbol{\beta}$ in SPAA is also lower than that in RLS or MWLS. This is also reflected in a smaller variance in prediction for SPAA as compared to either RLS or MWLS. The variance in the prediction for SPAA, RLS and MWLS is 0.167, 0.173 and 0.187 respectively. The number of updates can be further controlled by appropriately tuning the value of the threshold/sensitivity parameter, e. The effect of the sensitivity/threshold parameter is illustrated by Figure 4.10 which displays the parameter estimates for the regression vector of the linear model in Eqn. (4.32). Though small at 1.5 %, it is still significant since it is a percentage value and the window size at 30 is large.

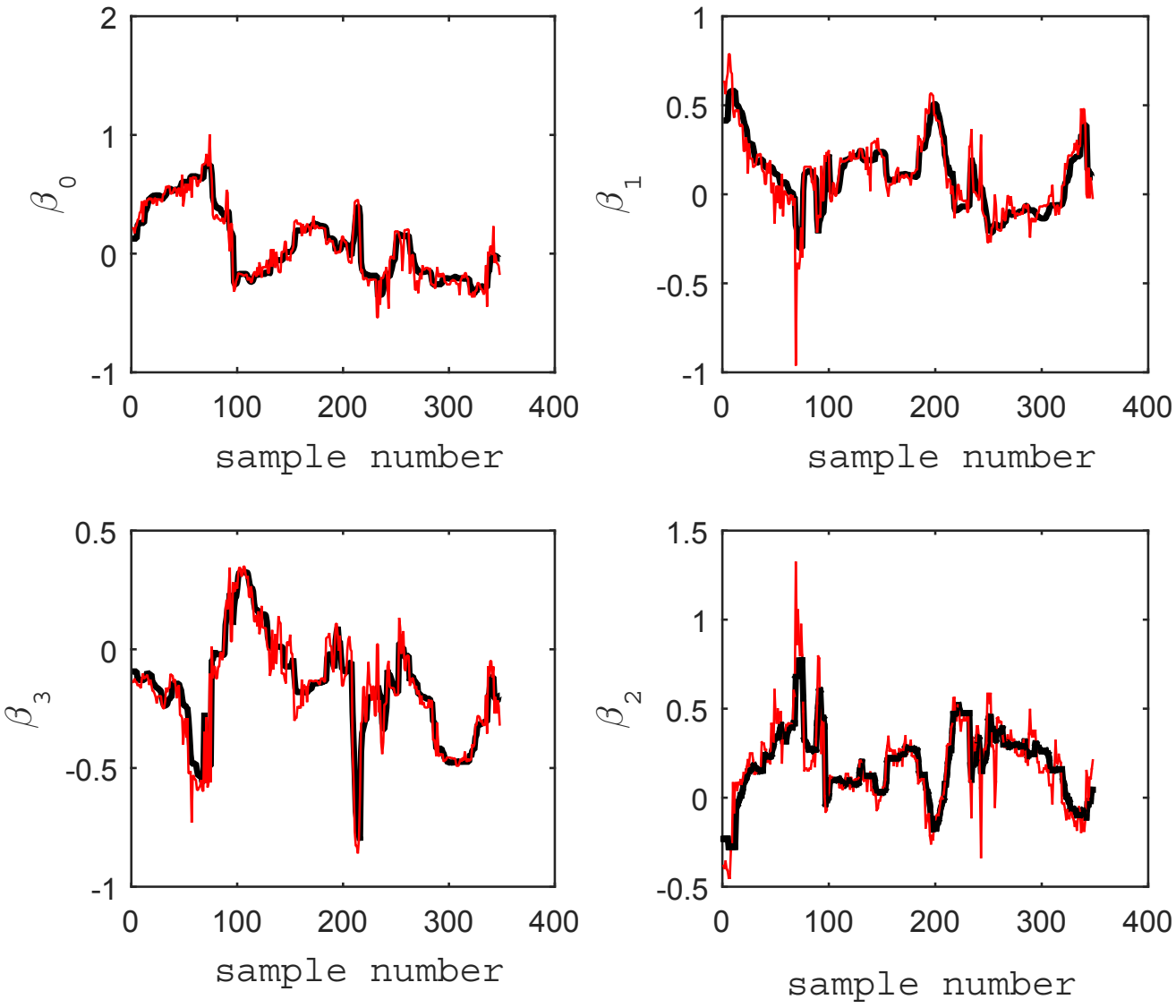

Figure 4.10: Clockwise from top left, parameter estimates for the constant term, flowrate, temperature and pressure respectively. - SPAA($e = 0$), - SPAA($e = 1.5$)

4.5 Conclusion

In this chapter, a new method called SPAA, which improves from an existing adaptive linear regression algorithm, OPAA, to make it more robust and suitable for practical applications, has been proposed. Compared to OPAA, RLS and MWLS, SPAA is more robust and follows a cautious parameter update strategy. Also, the SPAA

framework is quite flexible and general. OPAA and moving window LAD are realized from it at specific values of the tuning parameters and a number of variations are possible with little or no additional computational complexity. The advantages of the method are highlighted by application to an industrial and numerically simulated data set.

Chapter 5

Conclusions

5.1 Summary

This thesis focuses on the development of models for building soft sensors for prediction applications. Although soft sensors have, over the years, received considerable research attention, there are still many challenges faced during industrial applications. The major theme addressed in this thesis is the adaptability of soft sensors so that sustained performance is achieved without the need for model re-training after fixed time intervals. Secondly, simple linear models are considered. The low computational cost associated with such models gives them an advantage over the other more complex and heavier model structures.

Chapter 1 briefly introduced the need and justification for developing soft sensors in the first place. In Chapter 2, the issue of handling nonlinear and time varying systems simultaneously under the JIT modeling framework was addressed. Since the similarity criteria are typically based on space, it may happen that old samples get large weights in the local model. This will decrease the model performance if the system is time varying. Therefore a novel similarity criterion, which takes account of time as an additional variable, was introduced to calculate sample weights. Further, besides the offline strategy of determining the bandwidth parameters, an adaptive method was proposed as an alternative. The new method, based on minimizing the leave-one-out cross validation, finds the bandwidth parameters adaptively with respect to each query. Advantages of the methods were illustrated by application to numerically simulated and industrial NIR data. It was found that the proposed methods outperformed the traditional Euclidean space based JIT models as well as

RLWPLS, an existing method to deal with time varying nonlinear systems under the JIT framework.

Chapter 3 again dealt with JIT based soft sensors. The shortcomings of the existing distance-angle similarity metric were highlighted and an improved weighing scheme formulated. Secondly, point-based bandwidth selection strategy, where a bandwidth is stored with every historical data, was proposed to efficiently utilize available data. Since the point-based method is offline, the increase in computation cost associated with it is not a major concern. Results obtained from application to simulated and experimental laboratory data justified the claims made. Besides increase in prediction accuracy, clearer interpretation of results and an intuitive basis for parameter selection were observed as advantages of the new methods.

Chapter 4 introduced an existing adaptive linear regression algorithm, called OPAA. OPAA was then improved to make it suitable for industrial applications and the new algorithm thus developed was called SPAA. Linear regression algorithms such as the moving window least squares and recursive least squares are not robust due to the squared error cost function and are susceptible to minor process disturbances. SPAA, which uses an absolute deviation cost function and cautious parameter update strategy was shown to overcome these drawbacks. The methods were then applied to a numerical example and data from a de-propanizer column. Results indicated that SPAA performed better in the presence of outliers and had fewer parameter updates in comparison to the other algorithms.

5.2 Recommendations for future work

The first two chapters focus on improving the similarity criterion to increase the accuracy of JIT based modeling. One common element among the two is the structure of the similarity/weight function. The first step is to extend the normal input space to include metrics/new variables that give additional information regarding the system. The additional variables can be either the age of the sample, as in Chapter 1, to deal with time varying issue, or the angle, as in Chapter 2, to deal with the dynamics of the process. The weight is then calculated by simply taking the Euclidean distance between any two input variables thus modified. In this work, only two additional

variables, time and angle have been considered. However, other metrics such as correlation, used by Fujiwara et al. [23], could also be incorporated along similar lines. Secondly, since the similarity criterion is evaluated based on distance in the input space only, and squared error cost functions are used in the local models, they are susceptible to outlying output, y, values. Therefore, use of robust local models to address this issue can be an area for potential future work.

With regards to SPAA, developed in Chapter 3, it is noted that it is robust to outliers in the output space but is sensitive to large leverage points, i.e., outliers in the predictor variable [55, 56]. To handle outliers in the input variables, robust distance measures such as MCD or MVE [57] can be explored. More guidelines regarding the use of the SPAA variants can be established. Finally, the window and threshold parameters used are global values obtained through offline optimization. Exploring adaptive estimation techniques for these parameters is another possible research direction.

Bibliography

[1] George Cybenko. Just-in-time learning and estimation. *Nato Asi Series F Computer and Systems Sciences*, 153:423–434, 1996.

[2] Andrew W Moore, Christopher G Atkeson, and Stefan A Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[3] Stefan Schaal, Christopher G Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[4] S Joe Qin, Hongyu Yue, and Ricardo Dunia. Self-validating inferential sensors with application to air emission monitoring. *Industrial & Engineering Chemistry Research*, 36(5):1675–1685, 1997.

[5] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 33(4):795–814, 2009.

[6] Shima Khatibisepehr, Biao Huang, and Swanand Khare. Design of inferential sensors in the process industry: A review of bayesian methods. *Journal of Process Control*, 23(10):1575–1596, 2013.

[7] Petr Kadlec, Ratko Grbić, and Bogdan Gabrys. Review of adaptation mechanisms for data-driven soft sensors. *Computers & chemical engineering*, 35(1):1–24, 2011.

[8] Manabu Kano, Koichi Miyazaki, Shinji Hasebe, and Iori Hashimoto. Inferential control system of distillation compositions using dynamic partial least squares regression. *Journal of Process Control*, 10(2):157–166, 2000.

[9] Greger Andersson, Peter Kaufmann, and Lars Renberg. Non-linear modelling with a coupled neural network - pls regression system. *Journal of Chemometrics*, 10(5-6):605–614, 1996.

[10] Dong Eon Lee, Ji-Ho Song, Sang-Oak Song, and En Sup Yoon. Weighted support vector machine for quality estimation in the polymerization process. *Industrial & engineering chemistry research*, 44(7):2101–2105, 2005.

[11] Araby I Abdel-Rahman and Gino J Lim. A nonlinear partial least squares algorithm using quadratic fuzzy inference system. *Journal of Chemometrics*, 23(10):530–537, 2009.

[12] Cheng Cheng and Min-Sen Chiu. A new data-based methodology for nonlinear process modeling. *Chemical Engineering Science*, 59(13):2801–2810, 2004.

[13] Zhiqiang Ge and Zhihuan Song. A comparative study of just-in-time-learning based methods for online soft sensor modeling. *Chemometrics and Intelligent Laboratory Systems*, 104(2):306–317, 2010.

[14] Sanghong Kim, Manabu Kano, Shinji Hasebe, Akitoshi Takinami, and Takeshi Seki. Long-term industrial applications of inferential control based on just-in-time soft-sensors: economical impact and challenges. *Industrial & Engineering Chemistry Research*, 52(35):12346–12356, 2013.

[15] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106, 2004.

[16] Xueqin Liu, Uwe Kruger, Tim Littler, Lei Xie, and Shuqing Wang. Moving window kernel pca for adaptive monitoring of nonlinear processes. *Chemometrics and Intelligent Laboratory Systems*, 96(2):132–143, 2009.

[17] Chunfu Li, Hao Ye, Guizeng Wang, and Jie Zhang. A recursive nonlinear pls algorithm for adaptive nonlinear process modeling. *Chemical engineering & technology*, 28(2):141–152, 2005.

[18] Yi Liu, Haiqing Wang, and Ping Li. Modeling of fermentation processes using online kernel learning algorithm. In *Proceedings of IFAC World Congress*, pages 9679–9684, 2008.

[19] Manabu Kano, Sanghong Kim, Ryota Okajima, and Shinji Hasebe. Industrial applications of locally weighted pls to realize maintenance-free high-performance virtual sensing. In *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, pages 545–548. IEEE, 2012.

[20] Shiqi Zheng, Xiaoqi Tang, Bao Song, Shaowu Lu, and Bosheng Ye. Stable adaptive pi control for permanent magnet synchronous motor drive based on improved jitl technique. *ISA transactions*, 52(4):539–549, 2013.

[21] Hiroshi Nakagawa, Takahiro Tajima, Manabu Kano, Sanghong Kim, Shinji Hasebe, Tatsuya Suzuki, and Hiroaki Nakagami. Evaluation of infrared-reflection absorption spectroscopy measurement and locally weighted partial least-squares for rapid analysis of residual drug substances in cleaning processes. *Analytical chemistry*, 84(8):3820–3826, 2012.

[22] Sanghong Kim, Manabu Kano, Hiroshi Nakagawa, and Shinji Hasebe. Estimation of active pharmaceutical ingredients content using locally weighted partial

least squares and statistical wavelength selection. *International journal of pharmaceutics*, 421(2):269–274, 2011.

[23] Koichi Fujiwara, Manabu Kano, Shinji Hasebe, and Akitoshi Takinami. Soft-sensor development using correlation-based just-in-time modeling. *AIChE Journal*, 55(7):1754–1765, 2009.

[24] Ziyi Wang, Tomas Isaksson, and Bruce R Kowalski. New approach for distance measurement in locally weighted regression. *Analytical Chemistry*, 66(2):249–260, 1994.

[25] Mulang Chen, Swanand Khare, and Biao Huang. A unified recursive just-in-time approach with industrial near infrared spectroscopy application. *Chemometrics and Intelligent Laboratory Systems*, 135:133–140, 2014.

[26] Brian McWilliams and Giovanni Montana. A press statistic for two-block partial least squares regression. In *Computational Intelligence (UKCI), 2010 UK Workshop on*, pages 1–6. IEEE, 2010.

[27] Raymond H Myers. *Classical and Modern Regression With Applications*. Pws-Kent, Boston, MA, 1990.

[28] Gavin C Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1661–1668. IEEE, 2006.

[29] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[30] Luigi Fortuna, Salvatore Graziani, Alessandro Rizzo, and Maria Gabriella Xibilia. *Soft sensors for monitoring and control of industrial processes*. Springer, London, 2007.

[31] Sanghong Kim, Ryota Okajima, Manabu Kano, and Shinji Hasebe. Development of soft-sensor using locally weighted pls with adaptive similarity measure. *Chemometrics and Intelligent Laboratory Systems*, 124:43–49, 2013.

[32] Hiroyasu Shigemori, Manabu Kano, and Shinji Hasebe. Optimum quality design system for steel products through locally weighted regression model. *Journal of Process Control*, 21(2):293–301, 2011.

[33] J Duane Morningred, Bradley E Paden, Dale E Seborg, and Duncan A Mellichamp. An adaptive nonlinear predictive controller. In *American Control Conference, 1990*, pages 1614–1619. IEEE, 1990.

[34] Yuri Shardt, Ruben Gonzalez, and Aditya Tulsyan. *ChE 662: Process Identifcation Experimental Lab Manual*. CME Departmment, University of Alberta, AB, 2012.

[35] Seppo Pynnönen and Timo Salmi. A report on least absolute deviation regression with ordinary linear programming. *Finnish Journal of Business Economics*, 43(1):33–49, 1994.

[36] Subhash C Narula and John F Wellington. The minimum sum of absolute errors regression: A state of the art survey. *International Statistical Review/Revue Internationale de Statistique*, pages 317–326, 1982.

[37] Peter J Huber. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, pages 799–821, 1973.

[38] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[39] Jin Jiang and Youmin Zhang. A revisit to block and recursive least squares for parameter estimation. *Computers & Electrical Engineering*, 30(5):403–416, 2004.

[40] Babatunde A Ogunnaike. *Random phenomena: fundamentals of probability and statistics for engineers*. CRC Press, Boca Raton, Fl, 2010.

[41] Lennart Ljung and Torsten Söderström. *Theory and practice of recursive identification*. MIT Press, Cambridge, MA, 1983.

[42] Lennart Ljung. *System Identification: Theory for the user*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[43] Petr Kadlec. On robust and adaptive soft sensors. *Bournemouth University, School of Design, Engineering & Computing, Bournemouth, UK*, 2009.

[44] Robert J Vanderbei. *Linear Programming: Foundations and Extensions*. Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, 2001.

[45] Harvey M Wagner. Linear programming techniques for regression analysis. *Journal of the American Statistical Association*, 54(285):206–212, 1959.

[46] RL Branham Jr. Alternatives to least squares. *The Astronomical Journal*, 87:928–937, 1982.

[47] Robert Blattberg and Thomas Sargent. Regression with non-gaussian stable disturbances: Some sampling results. *Econometrica*, 39(3):501–510, 1971.

[48] Terry E Dielman. Least absolute value regression: recent contributions. *Journal of statistical computation and simulation*, 75(4):263–286, 2005.

[49] Steven P Ellis. Instability of least squares, least absolute deviation and least median of squares linear regression. *Statistical Science*, pages 337–344, 1998.

[50] M Planitz and J Gates. Strict discrete approximation in the l1 and l∞ norms. *Applied statistics*, pages 113–122, 1991.

[51] Stephen Portnoy and Roger Koenker. The gaussian hare and the laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 1997.

[52] Walter D Fisher. A note on curve fitting with minimum deviations by linear programming. *Journal of the American Statistical Association*, 56(294):359–362, 1961.

[53] Avi Giloni, Jeffrey S Simonoff, and Bhaskar Sengupta. Robust weighted lad regression. *Computational statistics & data analysis*, 50(11):3124–3140, 2006.

[54] Tomas Cipra. Robust exponential smoothing. *Journal of Forecasting*, 11(1):57, 1992.

[55] Yadolah Dodge. Lad regression for detecting outliers in response and explanatory variables. *Journal of multivariate analysis*, 61(1):144–158, 1997.

[56] Mia Hubert and Peter J Rousseeuw. Robust regression with both continuous and binary regressors. *Journal of Statistical Planning and Inference*, 57(1):153–163, 1997.

[57] Olcay Arslan. Weighted lad-lasso method for robust parameter estimation and variable selection in regression. *Computational Statistics & Data Analysis*, 56(6):1952–1965, 2012.

Appendix A

Figures displaying the tracking performance of ARX model parameters a_1, a_2 & a_3 for Test Cases I & II, Section 4.4.1, Chapter 4.

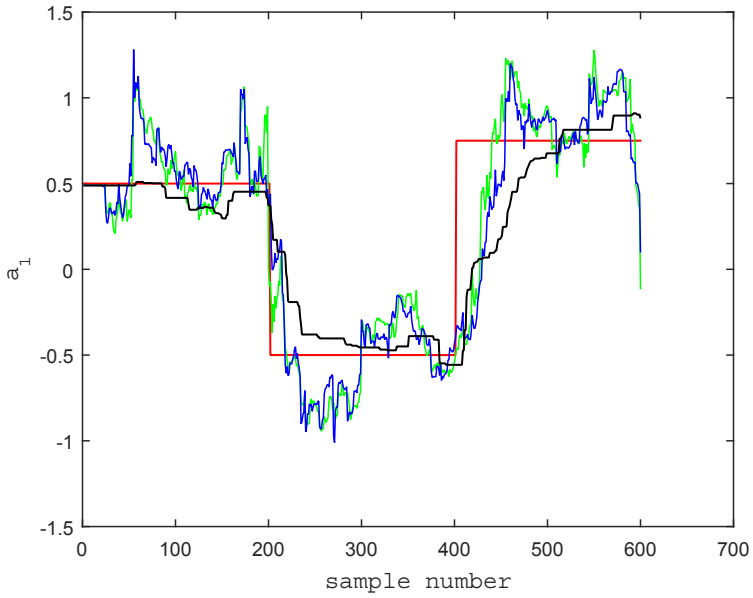

Figure 1: Tracking a_1, Test Case I. - True, - RLS, - MWLS, - SPAA

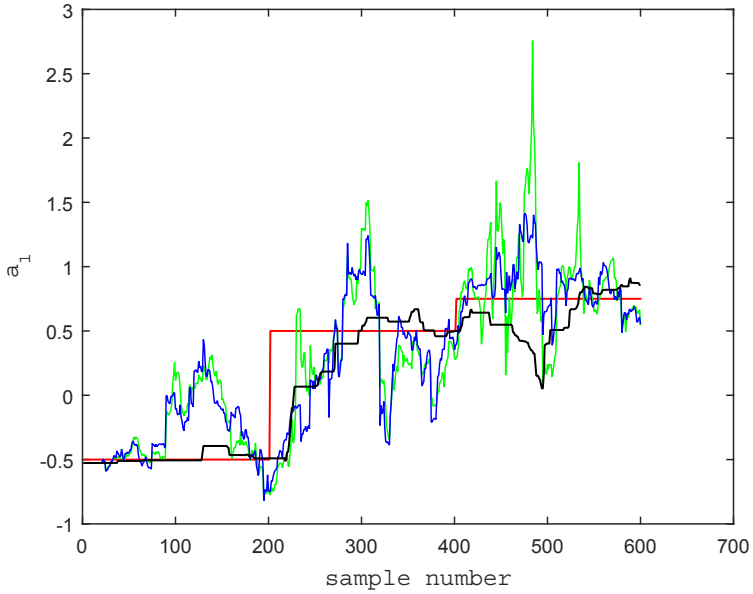

Figure 2: Tracking a_1, Test Case II. - True, - RLS, - MWLS, - SPAA

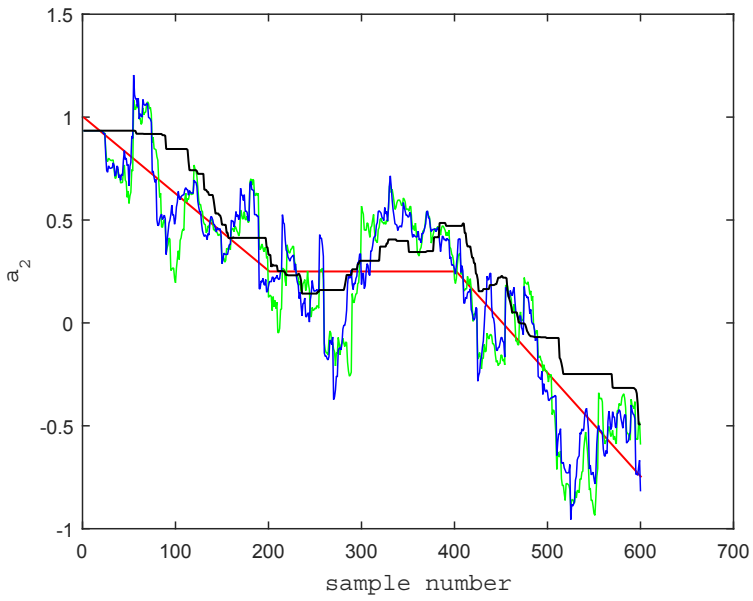

Figure 3: Tracking a_2, Test Case I. - True, - RLS, - MWLS, - SPAA

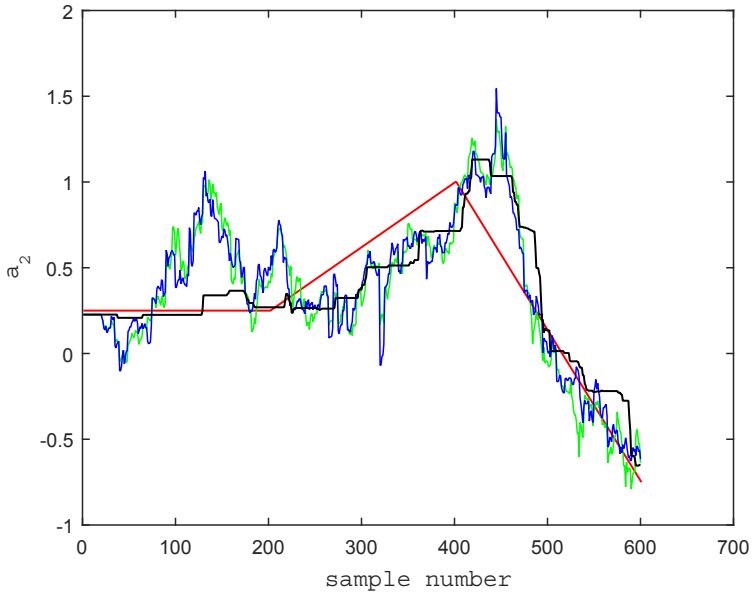

Figure 4: Tracking a_2, Test Case II. - True, - RLS, - MWLS, - SPAA

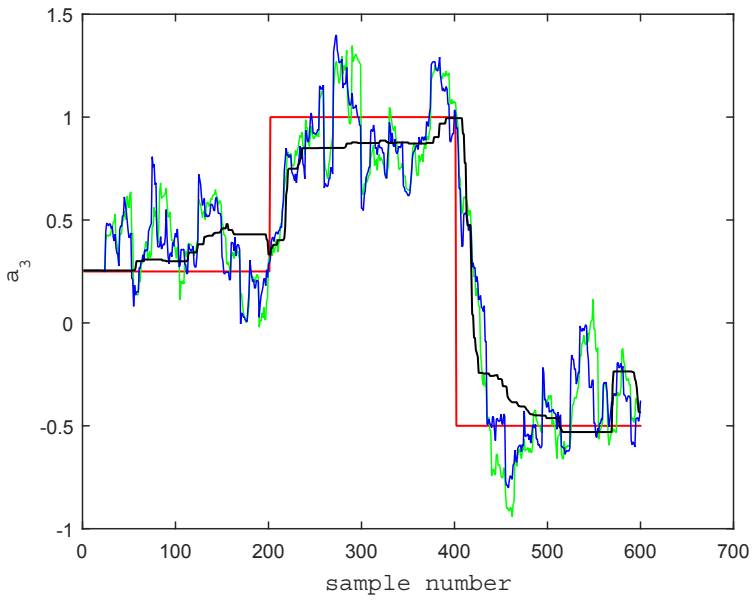

Figure 5: Tracking a_3, Test Case I. - True, - RLS, - MWLS, - SPAA

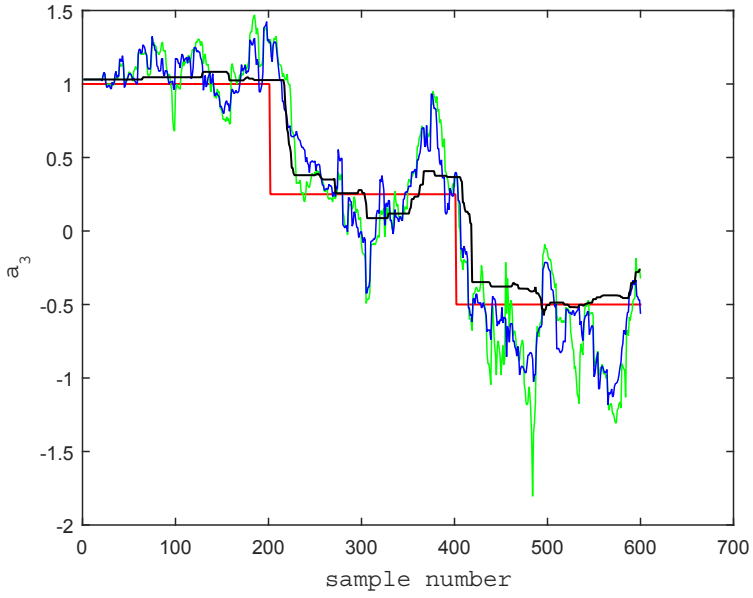

Figure 6: Tracking a_3, Test Case II. - True, - RLS, - MWLS, - SPAA

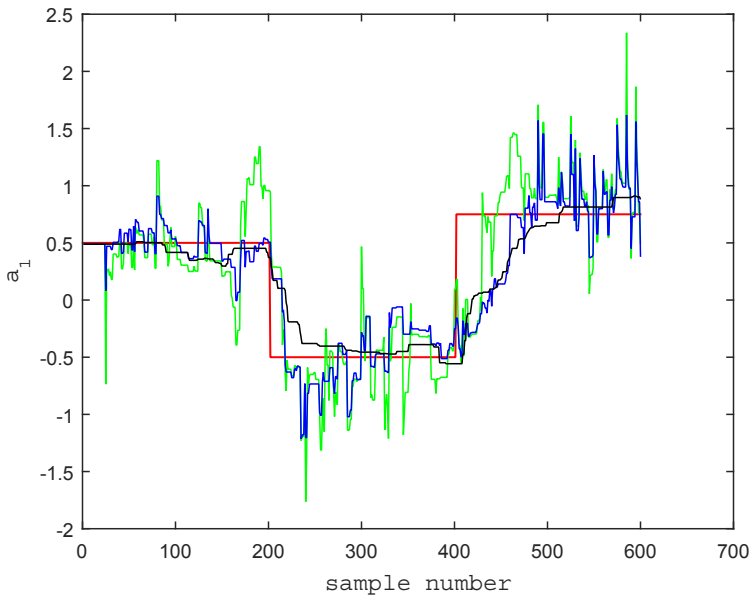

Figure 7: Tracking a_1, Test Case I. - True, - OPAA-I, - OPAA, - SPAA

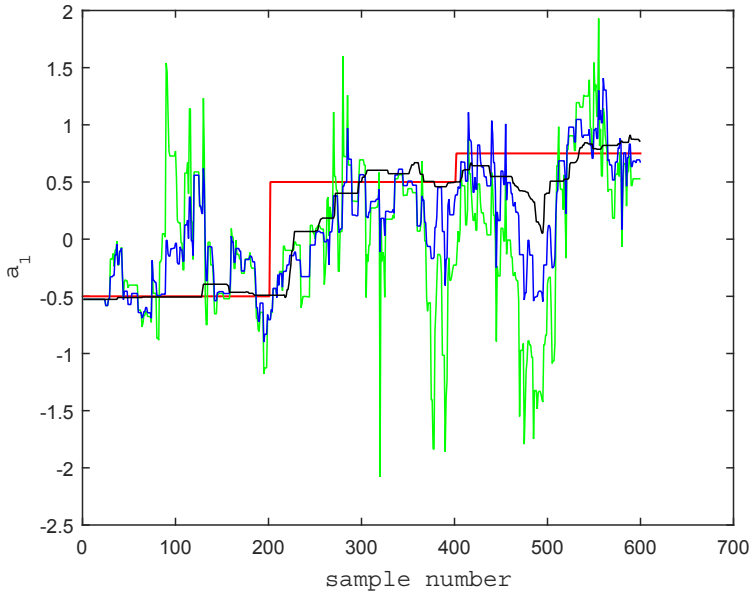

Figure 8: Tracking a_1, Test Case II. - True, - OPAA-I, - OPAA, - SPAA

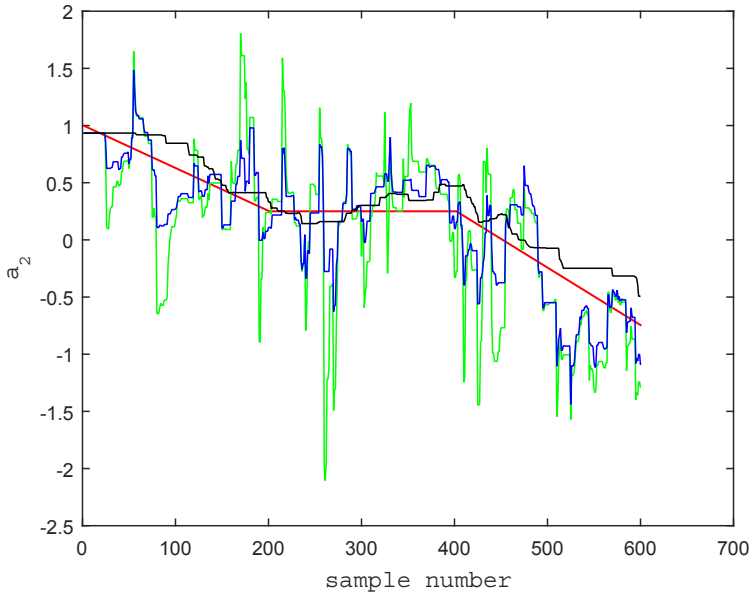

Figure 9: Tracking a_2, Test Case I. - True, - OPAA-I, - OPAA, - SPAA

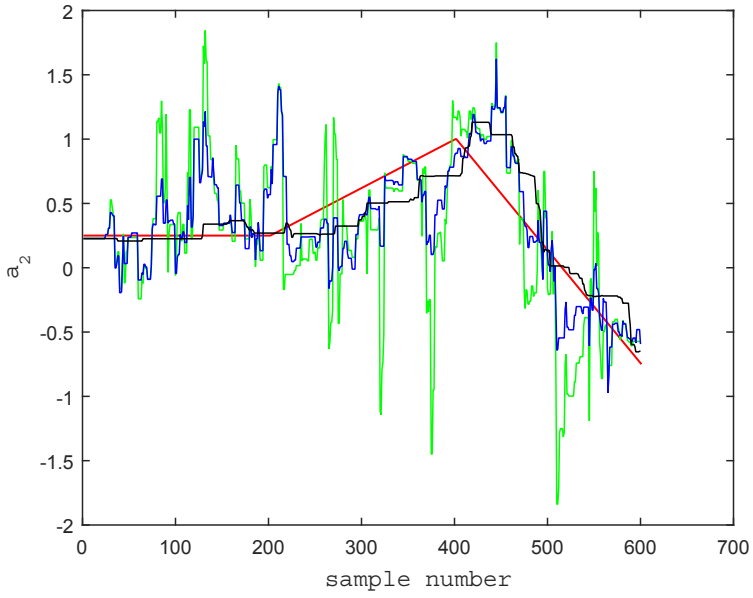

Figure 10: Tracking a_2, Test Case II. - True, - OPAA-I, - OPAA, - SPAA

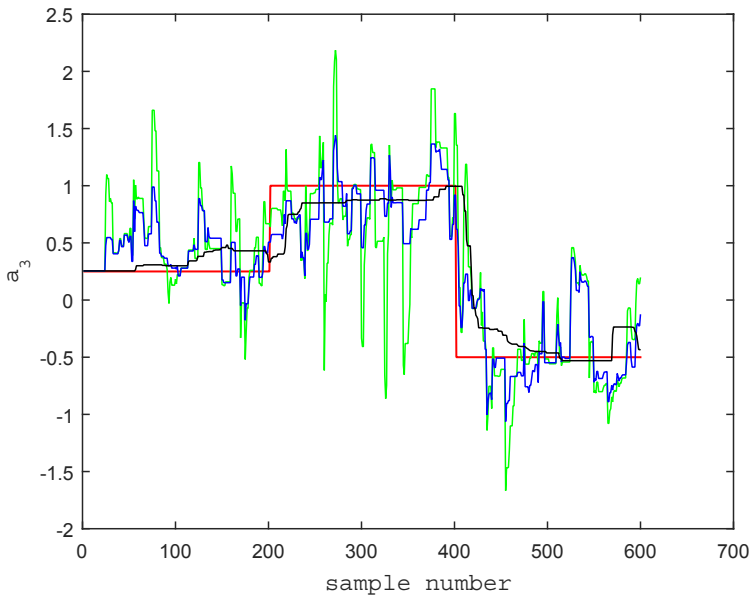

Figure 11: Tracking a_3, Test Case I. - True, - OPAA-I, - OPAA, - SPAA

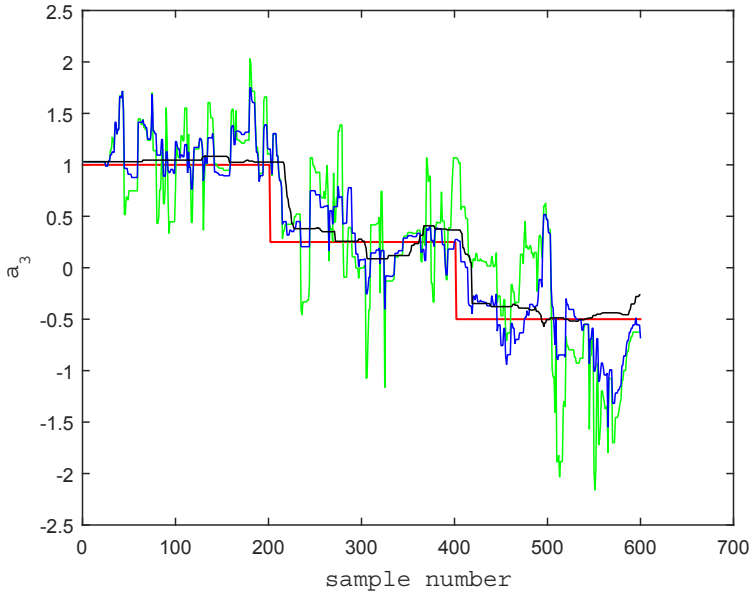

Figure 12: Tracking a_3, Test Case II. - True, - OPAA-I, - OPAA, - SPAA