Fault identification utilizing hybrid modelling based feature extraction models

by

Fabian Ethan Ferreira

Thesis presented in partial fulfilment
of the requirements for the Degree

of

MASTER OF ENGINEERING
(CHEMICAL ENGINEERING)

in the Faculty of Engineering
at Stellenbosch University

Supervisor

Dr. J.T. Cripwell

Co-Supervisor(s)

Prof. T.M. Louw

April 2022

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2022

ABSTRACT

Fault detection and identification models are critical in process monitoring and control as the models are essential in maintaining normal operating conditions. Fault identification models identifies the types of fault which occur in a process and the cause of the fault which allows corrective measures to be applied. Many fault identification models operate by identifying a process fault once a fault detection model has detected the presence of a fault.

Fault identification is posed as a multiclass classification problem with each class corresponding to a fault case with a normal operation class introduced to account for the fault detection aspect. A one-vs-one multiclass support vector machine (SVM) classifier is proposed as the fault identification model. A model parameter estimation method was proposed to improve the performance of the fault identification model. The parameter estimation behaves as a feature extraction method.

Hybrid modelling is identified as a possible model parameter estimation method. Hybrid modelling combines first-principle models and data-based models. The data-based models are trained to estimate the model parameters based on incoming process data. The data-based models considered are partial least squares regression (PLS), dynamic PLS, and recursive PLS models.

A non-isothermal jacketed continuous stirred tank reactor model is developed as a test case model with a catalyst deactivation fault, inlet concentration fault and heat transfer fault applied to the model. The fault identification models are trained using process data corresponding to a catalyst deactivation-inlet concentration fault pair and catalyst deactivation–heat transfer fault pair. The performance of the fault identification models is compared using the sensitivity and specificity measures.

The performance of fault identification models using a standard SVM and kernel SVM with a radial basis function kernel were compared. The kernel SVM showed similar performance to the SVM for the catalyst deactivation fault and heat transfer fault with sensitivity values of 0.684 ± 0.044 and 0.752 ± 0.067, and a shorter training time than the standard SVM model. When the performance of the classifiers incorporating non-linearly regressed model parameters were evaluated by identifying the catalyst deactivation fault and heat transfer fault it was found that the standard SVM model using the regressed parameters had higher sensitivities (0.686 ± 0.042, 0.811 ± 0.031) and specificities (0.989 ± 0.005, 0.968 ± 0.027) than the kernel SVM with sensitivities (0.633 ± 0.058, 0.734 ± 0.033) and specificities (0.974 ± 0.005, 0.924 ± 0.038) using the regressed parameters. When the performance of the hybrid fault identification models was evaluated, the standard SVM using dynamic PLS showed better performance than the other models with higher sensitivities (0.695 ± 0.041, 0.761 ± 0.056) and specificities (0.9800 ± 0.004, 0.949 ± 0.049). When the performance of all the models were compared it was found that the standard SVM using non-linearly regressed parameters was the best performing model.

The multiclass SVM approach has been shown a viable fault identification method. Implementing the model-based feature extraction method was shown to improve the performance of fault identification models. The SVM model using non-linearly regressed parameter estimates was found to be the best performing model. It is recommended that in future work the PLS models are replaced with another data-based model such as artificial neural networks.

OPSOMMING

Foutopsporing- en -identifikasiemodelle is krities in prosesmonitering en -beheer omdat die modelle essensieel is vir handhawing van normale bedryfskondisies. Foutidentifikasiemodelle identifiseer die tipe fout wat plaasvind in 'n proses en die oorsaak van die fout, wat die korrekte maatreëls toelaat om toegepas te word. Baie foutidentifikasiemodelle word bedryf deur 'n prosesfout te identifiseer sodra 'n foutopsporingsmodel die teenwoordigheid van 'n fout opgespoor het.

Foutidentifisering word voorgestel as 'n multiklas-klassifiseringsprobleem met elke klas wat korrespondeer met 'n foutgeval, met 'n normale bedryfklas voorgestel om die foutopsporingsaspek in berekening te bring. 'n Een-vs.-een multiklas ondersteuningvektormasjien (SVM) klassifiseerder word voorgestel as die foutidentifikasiemodel. 'n Modelparameterberamingsmetode is voorgestel om die doeltreffendheid van die foutidentifikasiemodel te verbeter. Die parameterberaming tree op as 'n kenmerkekstraksiemetode.

Hibried modellering is geïdentifiseer as 'n moontlike modelparameterberamingsmetode. Hibried modellering kombineer eerste-benaderingmodelle en datagebaseerde modelle. Die datagebaseerde modelle word opgelei om die modelparameters te beraam gebaseer op inkomende prosesdata. Die datagebaseerde modelle oorweeg is gedeeltelike kleinste kwadraatregressie (PLS), dinamiese gedeeltelike kleinste kwadraat-, en rekursiewe gedeeltelike kleinste kwadraatmodelle.

'n Nie-isotermiese ommantelde kontinue geroerde reaktormodel is ontwikkel as 'n toetsgevalmodel met 'n katalisator deaktiveringsfout, inlaatkonsentrasiefout en hitte-oordragfout wat toegepas is op die model. Die foutidentifikasiemodelle is opgelei deur prosesdata wat met 'n katalisator deaktiveringsinlaatkonsentrasiefoutpaar en katalisator deaktiveringshitte-oordragfoutpaar korrespondeer, te gebruik. Die doeltreffendheid van die foutidentifikasiemodelle was vergelyk deur die sensitiwiteit- en spesifisiteitmaatreëls te gebruik.

Die doeltreffendheid van foutidentifikasiemodelle is vergelyk deur 'n standaard SVM en kern SVM met 'n radiale basiesfunksie te gebruik. Die kern SVM het soortgelyke doeltreffendheid getoon as die SVM vir die katalisator deaktiveringsfout en hitte-oordragsfout met sensitiwiteitswaardes van 0.684±0.044 en 0.752±0.067, en 'n korter opleidingstyd as die standaard SVM-model. Toe die doeltreffendheid van die klassifiseerders wat nie-liniêre regressie modelparameters inkorporeer geëvalueer is deur die katalisator deaktiveringsfout en hitte-oordragsfout te identifiseer, is dit gevind dat die standaard SVM-model wat die parameters gebruik, hoër sensitiwiteite (0.686±0.042, 0.811±0.031) en spesifisiteit het (0.989±0.005, 0.968±0.027) as die kern SVM met sensitiwiteite (0.633±0.058, 0.734±0.033) en spesifisiteit (0.974±0.005, 0.924±0.038) wat die parameters gebruik. Toe die doeltreffendheid van die hibriede foutidentifiseringmodelle geëvalueer is, het die standaard SVM wat dinamiese PLS gebruik beter doeltreffendheid getoon as die ander modelle met hoër sensitiwiteite (0.695±0.041, 0.761±0.056) en spesifisiteit (0.9800±0.004, 0.949±0.049). Toe die doeltreffendheid van al die modelle vergelyk is, is dit gevind dat die standaard SVM wat nie-liniêre regressie parameters gebruik, die beste presteer het.

Die multiklas SVM-benadering is getoon as 'n uitvoerbare foutidentifikasiemetode. Implementering van die model-gebaseerde kenmerkekstraksiemetode is bewys om die doeltreffendheid van foutidentifikasiemodelle te verbeter. Die SVM-model wat nie-liniêre regressie parameter beraminge gebruik het, is gevind om die beste presterende model te wees. Dit word aanbeveel dat toekomstige werk die PLS-modelle vervang met 'n ander datagebaseerde model soos kunsmatige neurale netwerke.

ACKNOWLEDGEMENTS

I wish to thank the following people in particular for their contributions to the project without whom the project would not have successful:

- Dr Jamie Cripwell, for his guidance and supervision throughout this project
- Professor Tobi Louw, for his insight and contributions during the project
- My parents, Denise and Norval Ferreira, for their continuous love and support
- My brother, Leslie Ferreira for his encouragement and support during the best and worst of times
- Bernard Harrison, for his kindness and encouragement during the project
- Sasol Pty (Ltd), for their financial support during the project

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

ACRONYM	DESCRIPTION
ANN	Artificial Neural Network
APE	Absolute Percentage Error
BBH	Basic Bagging Model
CSTR	Continuous Stirred Tank Reactor
DBM	Data Based Model
DPLS	Dynamic PLS
KNN	K-Nearest Neighbours
KSVM	Kernel Support Vector Machine
LDA	Linear Discriminant Analysis
MAPE	Mean Absolute Percentage Error
MM	Mechanistic Model
N.O.C	Normal Operating Conditions
NIPALS	Non-linear Iterative Partial Least Squares
NSBH	Negative correlation SVR bagging hybrid model
ODE	Ordinary Differential Equation
PCA	Principal Component Analysis
PLS	Partial least squares
R-SVM	SVM using non-linearly regressed parameters
R-KSVM	Kernel SVM using non-linearly regressed parameters

RBF	Radial Basis Function
RPLS	Recursive PLS
SVM	Support Vector Machine
SVR	Support Vector Regression

SYMBOL	DESCRIPTION
$\boldsymbol{a}$	Lagrange coefficient
$\boldsymbol{b}$	PLS regression coefficient
$\boldsymbol{b_i}$	Bias
$\boldsymbol{B}$	PLS regression coefficient matrix
$\boldsymbol{C_A}$	Concentration of A
$\boldsymbol{c_p}$	Heat capacity
$\boldsymbol{C}$	Covariance matrix
$\boldsymbol{d_k}$	Cumulative distance metric
$\boldsymbol{D}$	Euclidean distance metric
$\boldsymbol{E_a}$	Activation energy
$\boldsymbol{E}$	Error
$\boldsymbol{e}$	Model offset
$\boldsymbol{E}$	Input residual error matrix
$\boldsymbol{F}$	Flowrate
$\boldsymbol{F}$	Output residual matrix
$\boldsymbol{\Delta H_r}$	Heat of reaction

k_0	Pre—exponential factor
k	Kernel transformation
K	Gram matrix
l	Support Vector Machine target label
P_t	Process parameter at current time step
p	Polynomial degree
P	Input loading matrix
p	Input loading vector
Q	Output loading matrix
q	Output loading vector
R	Universal gas constant
r	Residual vector
t	Time
T	Temperature
T	Input score matrix
t	Input score vector
UA	Overall heat transfer coefficient
U	Output score matrix
u	Output score vector
V	Reactor volume
v	PLS output weight vector
w	PLS input weight vector
$\overline{x}$	Input matrix mean

$\boldsymbol{X}$	Input matrix
$\boldsymbol{Y}$	Output matrix
$\boldsymbol{y}$	Output vector
$\boldsymbol{z_f}$	Binary selection variable
$\boldsymbol{z}$	Neural network true output value

SYMBOL	DESCRIPTION
$\boldsymbol{\gamma}$	Radial basis function width parameter
$\boldsymbol{\lambda}$	Forgetting factor
$\boldsymbol{\Phi}$	Mapping function
$\boldsymbol{\rho}$	Density
$\boldsymbol{\omega}$	Neural network weight

1 INTRODUCTION

1.1 Fault detection and identification models

The detection and identification of faults is a major component of ensuring and maintaining normal operation on a plant (Miljković, 2011). By detecting the onset of faults in processes, operators can be alerted to the occurrence of the fault and can begin preparing the relevant corrective measures. The source of a fault needs to be correctly identified to ensure that the correct action is implemented to rectify the fault and mitigate the effects of the fault on the system. Data driven and statistical models have seen more use in fault detection and fault identification models in recent times (Miljković, 2011).

Fault detection models are responsible for detecting when processes deviate from normal operating conditions (Miljković, 2011). There are several possible fault detection methods used in industrial applications with many methods comparing the interactions between process measures and comparing the interactions to some set baseline performance. The baseline represents ideal process performance at normal operating conditions. If the absolute magnitude of error between the interactions and the baseline measure exceeds a set threshold the corresponding observations are identified as fault laden observations (Isermann, 2006). Fault detection models however only identify when a process fault occurs and not what the type of fault or what the cause of the fault is. The need to identify the type and cause of the faults gave rise to the development of fault identification models (Isermann, 2006).

Fault identification models determine and classify process faults by comparing the process data to previously documented fault cases (Isermann, 2005). Fault identification can be posed as a multiclass classification problem where each class corresponds to a fault case. Fault identification models consists of some classification model such as support vector machine (Onel, Kieslich & Pistikopoulos, 2019) or artificial neural network (Heo & Lee, 2018) which acts as the fault identifier. Most of the fault identification models reviewed in this work operate by identifying the fault once a separate fault detection model has detected a fault. A dual fault detection and identification method is considered by introducing a class which represents normal operations conditions (Heo & Lee, 2018). The fault detection aspect is due to the model being able to distinguish between fault conditions and normal operation.

The performance of the fault identification models could be improved by introducing additional process information to the fault identifier. This was seen in the work done by Mid and Dua (Che Mid & Dua, 2017) in which first-principle model parameters were estimated and fed as inputs to a classification model. The additional information could take the form of features selected from process data. Onel et al. (Onel *et al.*, 2019) performed feature selection on process data before supplying the data to the fault identifier which increased the performance of the fault identification model. Thus, if feature selection can improve the performance of a fault identification model, then it is possible that feature extraction can exhibit a similar increase in performance. First-principle model parameters could be used as additional process information which can be used to improve the performance of the fault identification as was done by (Che Mid & Dua, 2017). A method of estimating the first-principle model parameters is found in hybrid modelling.

Hybrid modelling is best described as the use of data-based models in conjunction with first-principle based models to develop a model with improved accuracy and extrapolative capabilities (von Stosch, Oliveira, Peres & Feyo de Azevedo, 2014). The development of such models would make effective use of large process data reserves to improve the ability of models to describe real world process performance. A possible data based model which can be used is partial least squares regression (Wold, Ruhe, Wold & Dunn III, 1984) (Geladi & Kowalski, 1986). In the context of fault identification, the hybrid model would be responsible for estimating model parameters based on incoming process data. The first-principle model would be used to estimate an initial set of training model parameters which corresponds to a set of training process data. A data-based model is then trained to estimate the model parameters based on incoming process data. The parameter estimates are then be fed to a classification model.

A support vector machine-based fault detection and identification model is thus proposed as a possible fault identification method. The model will make use of hybrid modelling where variations of the partial least squares regression model will be considered as the data-based model. The hybrid model will be used to estimate model parameters which will be fed to fault identifier along with process data.

1.2 Aim and objectives

Based on the on the points discussed above the aim of the project is to develop a fault classification model which incorporates hybrid modelling as model parameter estimator. The estimated model parameters are used in conjunction with the process data to perform the fault classification. The support vector machine (SVM) classifier is proposed as a fault identification model.

The aim is achieved through fulfilling the following objectives:

- Conduct a literature review to establish a fault identification approach
- Developing a test case model on which to test fault identification approach
- Develop a hybrid modelling approach for model parameter estimation with an appropriate data-based model
- Developing a hybrid-SVM model classification approach
- Evaluating identification performance of hybrid model to determine if the performance of the model is satisfactory

1.3 Thesis Layout

The thesis is organised in the following Chapters: Chapter 2 presents the theory used through the study. Chapter 3 contains a literature review in which applications of hybrid modelling, fault detection and fault identification models are discussed. Chapter 4 discusses the methodology and process of developing and implementing the hybrid models, fault identifiers and the finally the combined hybrid modelling based fault identification model. Chapter 4 also discusses the test model developed to evaluate the performance of the fault identification model. Chapter 5 discusses and illustrates the performance of the fault identification model. Chapter 6 discusses the conclusions drawn from the study and the recommendations based on the conclusions.

2 THEORY

The following chapter describes the fundamental theory and key equations which describe and form the basis of the work considered and the literature considered in this study. Each statistical method described in this chapter is accompanied by the key equations which describe and underly the method. The theory presented in this chapter can be referred to as needed, to provide context to the literature studied in this work and to understand the theoretical and mathematical basis of the work performed in the study.

2.1 Fault detection and identification

In process monitoring and control maintaining normal operating conditions (NOC) is crucial, however process faults do occur during plant operation. Faults can be defined as instances at which system operation deviates from standard operating conditions (Miljković, 2011). The identification and correction of these faults become key in maintaining standard operation. The need for eliminating these process faults as soon as they occur has given rise to the fields of fault detection and identification in process control and monitoring.

Fault detection can be defined as the process of determining whether a fault occurs within a process (Miljković, 2011). The occurrence of a fault can be detected by considering the dependencies and relationships between different process measurements and comparing these relationships at a particular time point to what has been deemed normal operation. Faults can occur due to sensor errors, errors in process controllers and actuators as well as errors within the process itself. Fault detection techniques and methods can generally be categorised into three subgroups namely: data driven techniques, process model-based techniques and knowledge-based techniques (Miljković, 2011).

Data driven techniques determines what can be deemed as normal process operation by only making use of historical process data (Miljković, 2011). Incoming process data is then compared to what is deemed normal operation and any significant deviations are identified as faulty instances. Some examples of data driven techniques are limit checking, principal component analysis (PCA) based fault detection, spectrum analysis and neural network based fault detection (Isermann, 2006). In limit checking fault detection some process measurement is compared to predetermined threshold values, once the threshold is exceeded a fault is detected

and an alarm is raised (Isermann, 2005). There are different forms of principal component analysis fault detection with one variation comprising of a process dataset subjected to PCA. Once projected to the principle components the data is then analysed to assess whether there is any significant change in the mean or variance of the data (Miljković, 2011) (Venkatasubramanian *et al.*, 2003). Other variations use Hotelling's T^2 statistic (Hotelling, 1931) in conjunction with PCA(Wierda, 1994)(Kourti & MacGregor, 1995). Spectrum analysis statistically estimates a spectrum within which process signals represents normal operation. Once any process signal falls outside the spectrum it is deemed as faulty data (Isermann, 2006). Spectrum analysis performs similarly to the limit checking technique however a key difference is that spectrum analysis extracts relevant characteristics from the incoming process signal through means such as Fourier transformations (Miljković, 2011). Limit checking however compares the output signals directly to some set limit. In neural network fault detection, a neural network is trained using both normal operation and faulty operation data. The output of the neural network states whether incoming process data is faulty or not (Heo & Lee, 2018).

Process model-based fault detection techniques compare actual process output to the output of a process model which describes the normal process operation (Isermann, 2005). An example of process model-based fault detection methods is the use of parity relationships to obtain output residuals (Gertler, 1997). The residuals obtained from the parity relations are then used in the fault detection. Another model-based technique is parameter estimation fault detection. In this technique unknown process model parameters are estimated using process input and output data. A relationship is obtained between the model parameters and the physical process parameters (Miljković, 2011). The estimated physical parameters are then compared to the true physical parameter values at normal operation and the residuals from the comparison are used for fault detection. Neural networks are also used in the model-based approach. A neural network can be trained as a non-linear process model to estimate process outputs. Residuals can then be estimated using parity equations (Schwarte & Isermann, 2002).

Knowledge-based fault detection techniques involve the use of expertise and experience to develop a fault detection method for a specific process (Venkatasubramanian *et al.*, 2003). Knowledge about the process is obtained from experienced plant operators and senior engineers on plant and is used to determine process production rules (Angeli, 2010). These rules are then used to determine when the process is operating under faulty conditions.

Fault identification (also referred to as fault diagnosis) is defined as determining the cause of the deviations from normal operation (Isermann, 2005). Where the aim of fault detection is determining when a fault occurs in a process and raising an alarm, fault identification determines the cause of the alarm being raised. As fault identification can be considered a classification problem, classification models are considered for use as fault identification models (Heo & Lee, 2018). There are several classification models available with some of the most commonly used models being k-nearest neighbours models, artificial neural network models and support vector machine classifiers(Onel *et al.*, 2019)(He & Wang, 2007)(Heo & Lee, 2018).

2.2 Curse of dimensionality

As fault detection methods are often trained on as much process data as is available, the training datasets become large with a large number of features. As the use of machine learning techniques in fault detection methods has become more common, when the techniques are applied to datasets with many features the curse of dimensionality becomes a concern. The curse of dimensionality is a common phenomenon in machine learning and mathematical modelling which was identified by Bellman (Bellman, 1961). It refers to the exponential increase in training data required for machine learning algorithms when applied to higher dimensional input datasets with many features (Bishop, 2006). This results in training data requirements which become unwieldy to effectively train machine learning models. Although this raises issues when applying machine learning algorithms to higher dimensional input datasets, these models can still be effectively applied. This is due to real world datasets often being able to be effectively expressed by some underlying features in a lower dimensional space which accounts for the variability in the original dataset (Bishop, 2006).

This gives rise to the use of dimensionality reduction methods. Dimensionality reduction methods are used to transform a dataset from a higher dimensional space to a lower dimensional space that describes the characteristics of the dataset in the original space (Van Der Maaten *et al.*, 2009). One subset of dimensionality reduction techniques is feature extraction models. Feature extraction models derives a new set of variables/features which describe the characteristics and variability of the original dataset in a lower dimensional space. There are various dimensionality reduction methods available such as principal component analysis or linear discriminant analysis (LDA) (Van Der Maaten *et al.*, 2009).

2.3 Introduction to hybrid modelling

Fault identification models often implement feature extraction (or feature selection) methods to improve the performance of classifiers used for fault identification (Onel *et al.*, 2019). A possible adaptation of feature extraction methods is be made by introducing concepts found in parameter estimation fault detection. Unknown model parameters can be estimated using process data and can be fed to a fault identifier. A possible estimation method which could be considered is hybrid modelling.

Hybrid modelling (also known as grey-box modelling) represents a more robust method of describing the behaviour of a system (von Stosch *et al.*, 2014). This is achieved through the combination of first-principle models and machine learning techniques with the machine learning techniques compensating for the shortcomings and errors associated with first-principle modelling. First-principle models are often not sufficient for applications in industry due to the constant variations in process conditions as well as aspects of the process which constantly need to be revised to ensure accuracy in the description of system behaviour (Bhutani *et al.*, 2006). In understanding hybrid modelling it is critical that one understands first-principle models and empirical modelling.

2.3.1 First-Principle Based modelling

Fundamental-based models (also known as first-principle or white box models) are models based on real world phenomena and physical laws which can be proven. The fundamental models are based on principles such as mass balances, energy balances and thermodynamic laws as these can be proven in the real world and are well documented and explained (von Stosch *et al.*, 2014). Fundamental models are also based on prior knowledge based on experience in running systems and are incorporated into first-principle based models. Developing and implementing fundamental models for specific processes can be time consuming as it may require performing a wide range of tests to obtain model parameters which ensure the best fit of the model to the process(Willis & von Stosch, 2017). The first-principle model might only be applicable to the process under certain conditions or constant variations in process results in an ill-fitting model. Based on these conditions more efficient method of utilising fundamental models are required with a possible solution being the use of hybrid modelling (Bhutani *et al.*, 2006).

2.3.2 Data-based modelling

Data-based models (also known as black box models) are models which are not based on physical laws but are data driven (Bhutani *et al.*, 2006). Examples of data-based models are artificial neural networks (ANN), partial least squares regression and support vector machine regression (SVR) to name a few (Bishop, 2006)(Geladi & Kowalski, 1986).

Training a data-based model refers to adjusting weights and parameters within the model to ensure that the model describes the training dataset. Ideally a data-based model is devised to describe the performance of a system and should not depend on the dataset used during training however this depends on the variance of the model (James *et al.*, 2017). More flexible models typically exhibit higher variance which means that small changes in training datasets lead to large changes in the models (James *et al.*, 2017). Bias refers to error incurred due to approximating a real-world problem via a mathematical model (James *et al.*, 2017). Ideally a model should display low variance and low bias however often one needs to develop a balance between variance and bias. Models with low bias tend to have higher variance and vice versa and this is known as the bias-variance trade off (James *et al.*, 2017).

Data-based models can be easier to define than first-principle models as they only require input and output data (Bhutani *et al.*, 2006). Thus, a combination of both modelling approaches would be ideal to incorporate the strongest aspects of both approaches. The desire for this combination gave rise to hybrid modelling.

2.3.3 Hybrid modelling

Hybrid modelling is defined as the combination of first-principle and data-based modelling approaches (von Stosch *et al.*, 2014). As hybrid modelling is a combination of parametric fundamental models and nonparametric data-based models it results in a hybrid semiparametric model which compensates for the short comings of the individual approaches. Based on this train of thought when presented with an equivalent dataset the hybrid model would provide more accurate predictions and would be developed in a shorter time than a fundamental model (von Stosch *et al.*, 2014). Hybrid models also improve on the performance of empirical models by being more robust and providing more accurate results when extrapolated to inputs beyond the scope of the initial training dataset whereas pure empirical models do not provide sufficient results beyond the scope of the initial training dataset.

When using hybrid modelling to describe the behaviour of a single system, large variations can occur in the nature of the modelling approaches used for the system. The variation is a result of not only the different fundamental and empirical models which can be used but also in the way the models can be integrated (von Stosch *et al.*, 2014). Variation also occurs due to the selection of which aspect of the system each class of model describes. Due to the variation the modelling approach followed depends heavily on the nature of the system as well as the nature of available data, the known and unknown aspects of the system.

When considering the implementation of hybrid models the most common model structures are series and parallel structures. Series structures (as shown in Figure 1 below) are one of the most popular structures of hybrid models (von Stosch *et al.*, 2014). Series structures work by feeding system inputs to an empirical model which produces an output which is fed to a fundamental model alongside the original input. Common uses of the empirical model is estimating model parameters or predicting model residuals (von Stosch *et al.*, 2014).

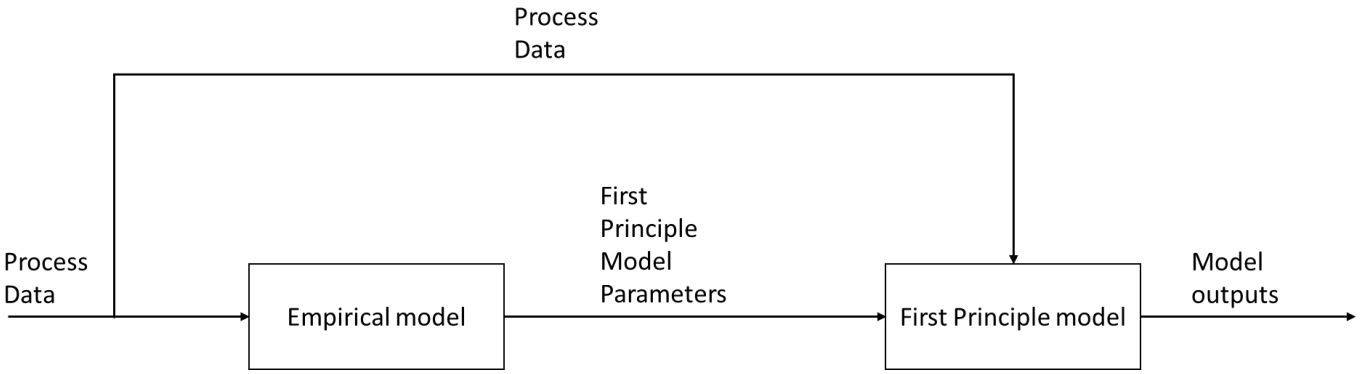

Figure 1: Series hybrid model structure

Series configurations are most effective when the fundamental model considered is accurate. The series structure uses incoming data to accurately predict the correct parameters for a fundamental model. These guessed parameters when used in fundamental models can improve the accuracy of the predictions. Series hybrid models can also be used when there are large data sets which relate to unknown parameters but with no direct correlation between them as the empirical model can be used to infer the parameter values based on the data set (von Stosch *et al.*, 2014).

An alternate series hybrid structure can be implemented as shown in Figure 2. In the alternate series configuration, the empirical model uses the output from the fundamental model to adjust process parameter values. These process parameters can be fed to the first-principle model, which can be used to adjust the process or can used in a different process model entirely (Aguiar & Filho, 2001). This configuration is not commonly used to describe the behaviour of chemical systems.

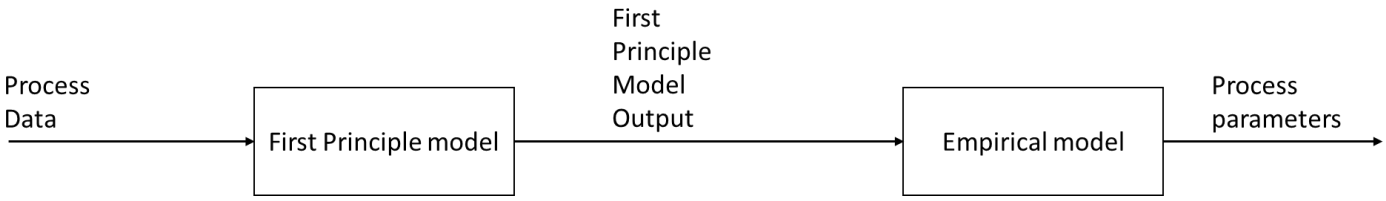

Figure 2: Alternate series structure

Parallel structures are another popular form of hybrid model structure as shown in Figure 3. In parallel structures the input data is fed simultaneously to both the empirical and fundamental models and the output of the respective models are then combined (von Stosch *et al.*, 2014)(Bhutani *et al.*, 2006). In this application the empirical model is used to predict the residuals of the model prediction to the true output values. The residuals used to train the empirical model are calculated by comparing the first-principle models outputs estimated using parameters which are thought to belong to the current process and comparing them to the true output values (Lee *et al.*, 2002). The residuals predicted using the incoming process data and the empirical model are then added to the output values estimated by the fundamental model. The aim of adding the predicted residuals to output values for the same observation is to improve the accuracy of the first-principle model prediction and to account for any mismatch between the predictions and real world values (Estrada-Flores *et al.*, 2006). Parallel models are applied when the fundamental models considered are not accurate in predicting the behaviour of the system (von Stosch *et al.*, 2014).

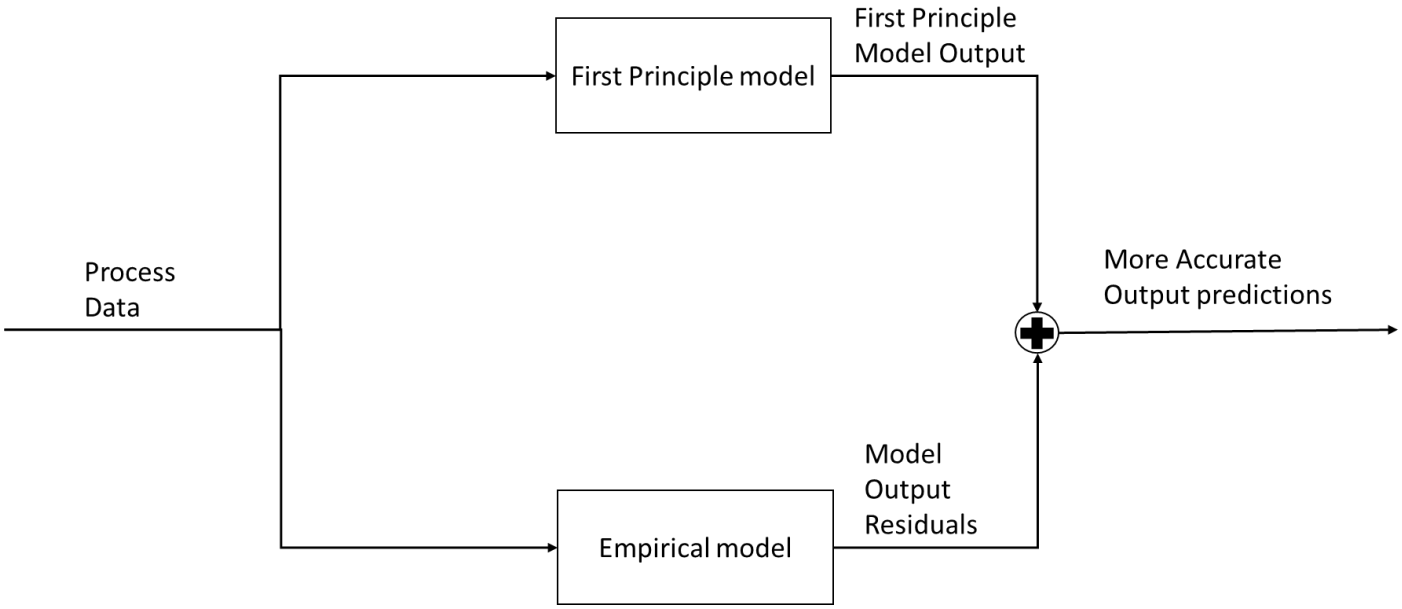

Figure 3: Parallel hybrid model structure

2.4 Principal component analysis

Principal component analysis (PCA) is a method of feature extraction and data analysis whereby the strength of interactions between variables of a given dataset are investigated (Wold, Esbensen & Geladi, 1987). The PCA algorithm represents a dataset as a linear combination of data variable projections (Shlens, 2014). PCA can also be used to screen the data for outliers as well. A dataset can be represented by a data matrix $\boldsymbol{X}$ with N rows which represent the observations and L columns which represent the features which can be analysed and decomposed by PCA.

Geometrically PCA can be thought of as a projection of a data matrix into some mathematical subspace. Thus, a data matrix $\boldsymbol{X}$ when projected into the subspace is represented as N objects (or data points) in a L-dimensional space. The direction of the projection axis is then described by projection matrix $\boldsymbol{P'}$ with the coordinates of each data point stored in matrix T (Wold *et al.*, 1987). The rows of the projection matrix, $\boldsymbol{P'}$, are known as loading vectors ($\boldsymbol{p}$) and the columns of the matrix $\boldsymbol{T}$ are known as score vectors ($\boldsymbol{t}$). The projection matrix geometrically represents directions in the subspace along which the datapoints vary the most (James *et al.*, 2017). Therefore, the projection matrix entries (the loading vectors) represent the directions of the principal component axes. The score values represents the magnitude of the observations when projected onto the principal component axis (Wold *et al.*, 1987). Thus, the scores

represents the location (or distance) of the observations with respect to the principal component axis. As an example, consider a 5x2 input data matrix ($\boldsymbol{X}$) where each column represents a variable/feature (thus $\boldsymbol{x_1}$ represents the first variable and $\boldsymbol{x_2}$ the second), and each row represent an observation, a coordinate in (x_1, x_2) space. The geometric projection of the dataset onto the first principal component is illustrated in Figure 4 below.

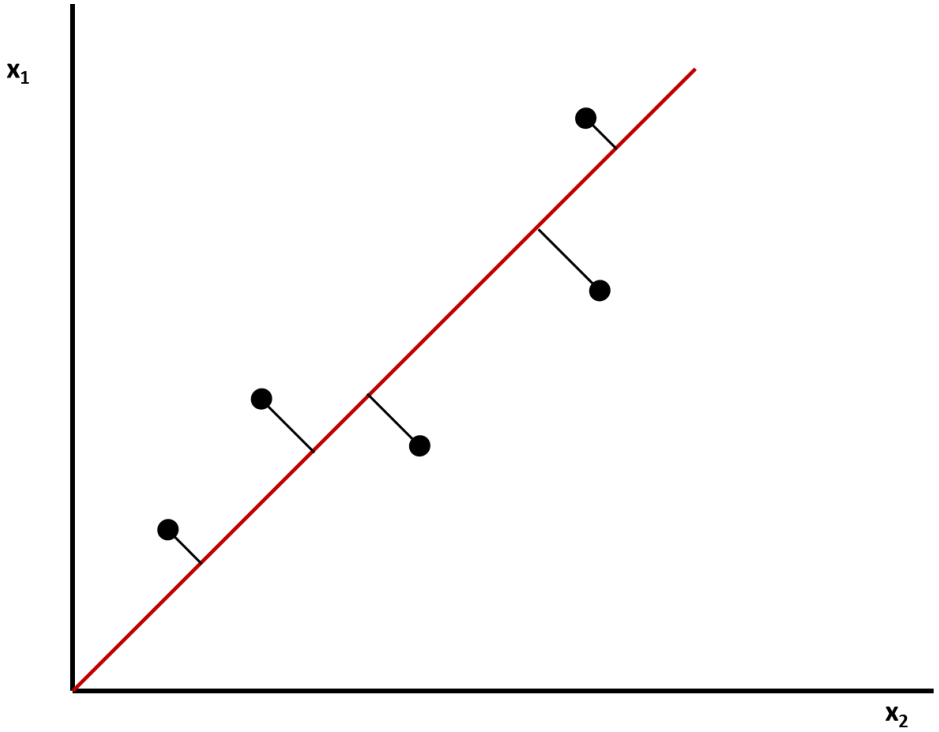

Figure 4: Projection of data onto the first principal component. The x- and y-axes represent the original features x_1 and x_2, respectively.

The red axis represents the principal component the direction of which is determined by the variability of the dataset. The projection of the data onto the principal component is the perpendicular distance between the data points and the principal component axis. These distances represent the scores of the observations with respect to the principal component as shown in the figure above.

Mathematically PCA can be defined as illustrated in Equation 1 (Wold *et al.*, 1987). In the definition $\bar{\boldsymbol{x}}$ represents the mean of the matrix and $\boldsymbol{X}$ and $\boldsymbol{E}$ represents residuals which are obtained from the difference between the PCA projection and the original data point. The loading vectors are normalized such that the sum of the mean square errors of the loading vectors are one (James *et al.*, 2017). When PCA is performed on a dataset the variables are

scaled to have zero mean thus eliminating the $\bar{\boldsymbol{x}}$ term. The input is also normally normalized such that the standard deviation each feature in the matrix is one.

$$\boldsymbol{X} = \bar{\boldsymbol{x}} + \boldsymbol{T}\boldsymbol{P}' + \boldsymbol{E} \qquad [\ 1\ ]$$

The principal component projections of the data matrix can be defined as a linear combination of the loading vectors and the datapoints(James *et al.*, 2017). The principal component projections are composed of score vectors, $\boldsymbol{t}$, with the entries in the score vector determined as shown in Equation 2 which illustrates how the first entry in the score vector is determined (James *et al.*, 2017). The subscript i indicates the i^{th} observation thus $\boldsymbol{t_{i1}}$ corresponds to the i^{th} entry in the score vector for the first principal component. The subscript k corresponds to the number of features in the dataset $\boldsymbol{X}$.

$$\boldsymbol{t_{i1}} = \boldsymbol{p_{11}x_{i1}} + \boldsymbol{p_{21}x_{i2}} + \cdots + \boldsymbol{p_{k1}x_{ik}} \qquad [\ 2\ ]$$

This allows the loading vectors to be determined by the optimisation equation shown in Equation 3. The loading vectors are also constrained such that they are orthogonal to one another. The setting of the loading vector values (and as such the direction of the principal component) is referred to as rotating the components (Shlens, 2014). This ensures that each principal component is uncorrelated and maximises the variance captured by the principal components (James *et al.*, 2017).

$$\max_{\boldsymbol{p}_{jk}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} \boldsymbol{p}_{jk}\, \boldsymbol{x}_{ij} \right)^2 \right\} with\ \sum_{j=1}^{m} \boldsymbol{p}_{jk}^2 = 1 \qquad [\ 3\ ]$$

The correlation of the variables is given by the covariance matrix obtained during the PCA. The covariance matrix ($\boldsymbol{C}$) is defined as shown in Equation 4. The diagonals of the covariance matrix represents the variance of the variables and the off-diagonal values represents the covariance between two variables (Shlens, 2014). If the covariance is zero it indicates that the variables are uncorrelated.

$$\boldsymbol{C} = \frac{1}{n}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} \qquad [\ 4\ ]$$

The covariance represents the linear relationship between two variables (Shlens, 2014). The magnitude of the covariance indicates the redundancy between variables with a high covariance indicating a high redundancy. By knowing the redundancy between variables one can perform dimensionality reduction by excluding redundant variables (Shlens, 2014). In PCA the principal components (i.e the loadings $\boldsymbol{p}$) are the eigenvectors of the covariance matrix (Shlens, 2014).

As seen PCA can be used as not only a dimensionality reduction method (which is achieved through feature extraction) but also allows a user to determine the covariance between process features. By removing redundant features, the PCA method shows clearer relationships between process features which are hidden by redundant features. When the percentage variance explained by each principal component is calculated it reveals how many process features are required to accurately describe the dataset. PCA has found use in fault detection methods such as the method shown by (Addo, 2019). PCA also forms the foundation of the partial least squares regression method.

2.5 Partial least squares regression

2.5.1 Standard partial least squares regression

Partial least squares (PLS) regression is regression technique which involves the projection of data to a mathematical space where the data can be described by a set of latent variables. The regression model is then based on and relates the input ($\boldsymbol{X}$) and output ($\boldsymbol{Y}$) observations in the latent space. The PLS regression obtains weight vectors $\boldsymbol{w}$ and $\boldsymbol{c}$ such that covariance is maximised as shown in Equations 5 and 6 (Rosipal, Trejo & Matthews, 2003).

$$\max_{\boldsymbol{w},\boldsymbol{c}}([cov(\boldsymbol{Xw},\boldsymbol{Yc})]^2) \qquad [\ 5\ ]$$

Subject to:

$$\boldsymbol{w}^T\boldsymbol{w} == 1 \qquad [\ 6\ ]$$

An advantage of the PLS algorithm is that it will find an entire set of weight vectors which are stored in weight matrices $\boldsymbol{W}$ and $\boldsymbol{C}$. The set of weight vectors are obtained such that the latent variables are orthogonal.

The projection of the data to the latent space is based on and resembles PCA (Geladi & Kowalski, 1986). The projection is performed on both the input, $\boldsymbol{X}$, and output, $\boldsymbol{Y}$, dataset and

the score matrices ($\boldsymbol{T}$ for $\boldsymbol{X}$ and $\boldsymbol{U}$ for $\boldsymbol{Y}$) are obtained for both datasets (James *et al.*, 2017). The equations resulting from the projection of the input (Equation 7) and output (Equation 8) matrices are shown below where $\boldsymbol{E}$ and $\boldsymbol{F}$ are residual matrices and $\boldsymbol{P}$ and $\boldsymbol{Q}$ are loading matrices (Geladi & Kowalski, 1986). The loading matrices contain the loading vectors where each vector represents the direction of a projection axis. The number of PLS components retained are indicated by the subscript h.

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}' + \boldsymbol{E} = \sum \boldsymbol{t_h}\boldsymbol{p_h}' + \boldsymbol{E} \quad [7]$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}' + \boldsymbol{F} = \sum \boldsymbol{u_h}\boldsymbol{q_h}' + \boldsymbol{F} \quad [8]$$

A linear relationship between the score vectors can be described by Equation 9 where $\boldsymbol{b}$ represents the regression coefficient and $\boldsymbol{r}$ represents the residual vector (Geladi & Kowalski, 1986). This linear relationship is also referred to as the inner relationship.

$$\boldsymbol{u}_i = \boldsymbol{t}_i b_i + \boldsymbol{r}_i \boldsymbol{A} \quad [9]$$

In Equation 9 the subscript i refers to which PLS component the vectors described i.e. the $\boldsymbol{X}$ score vector $\boldsymbol{t_1}$ is the score vector for the first principle component.

The inner relationship forms the foundation of the PLS regressions model by providing a relationship between the $\boldsymbol{X}$ scores and the $\boldsymbol{Y}$ scores. The regression coefficients $\boldsymbol{b}$ are adjusted iteratively to minimise the residual vector. As the inner relationship relates $\boldsymbol{t}$ and $\boldsymbol{u}$ it can be used to provide a relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ (Geladi & Kowalski, 1986). The direct relationship can be expressed by deriving a mixed PLS model which uses the inner relationship regression coefficients to estimate $\boldsymbol{Y}$ directly. The mixed PLS relationship model can be derived by combining the above equations and is shown in Equation 10 below where $\boldsymbol{B} = diag\{b_1, b_2 \dots b_n\}$ represents the regression coefficient matrix:

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{B}\boldsymbol{Q}' + \boldsymbol{F} \quad [10]$$

The objective of the PLS algorithm is to reduce the value of $|\boldsymbol{F}|$ as this produces more accurate estimations of $\boldsymbol{Y}$. This is achieved through an iterative algorithm in which the score vectors are used in unison to obtain more accurate score vectors (Geladi & Kowalski, 1986). There are

various algorithms available for a PLS regression such as the Non-linear Iterative Partial Least Squares (NIPALS) (Geladi & Kowalski, 1986) algorithm and the SIMPLS algorithm (S. De Jong, 1993). In this work the NIPALS algorithm is used as it the algorithm used in the recursive PLS variations.

Both $\boldsymbol{X}$ and $\boldsymbol{Y}$ datasets are projected and decomposed separately which results in a weak relationship between the projected matrices. In the NIPALS algorithm this potential shortcoming is addressed by relating the principal components of each decomposed data matrix (Geladi & Kowalski, 1986). The scores of the $\boldsymbol{X}$ and the $\boldsymbol{Y}$ matrices are exchanged during the regression algorithm where the projected $\boldsymbol{X}$ matrix gets the scores of the projected $\boldsymbol{Y}$ matrix and vice versa. This results in both input and output matrices obtaining information about each other strengthening the relationship between the two (Geladi & Kowalski, 1986).

The NIPALS algorithm is described as shown where $\boldsymbol{u}$ is the $\boldsymbol{Y}$ score vector, $\boldsymbol{t}$ is the $\boldsymbol{X}$ score vector, $\boldsymbol{w}$ is the weight vector, $\boldsymbol{p}$ and $\boldsymbol{q}$ are the $\boldsymbol{X}$ and $\boldsymbol{Y}$ loading vectors respectively:

1. Estimate a first guess for $\boldsymbol{Y}$ score vector $\boldsymbol{u}$. A normal first estimation would be using a vector from the $\boldsymbol{Y}$ output vector
2. Set weight vector such that $\boldsymbol{w^T = u^T X / u^T u}$
3. Normalise weight vector $\boldsymbol{w^T}_{new} = \boldsymbol{w^T_{old}} / \left\|\boldsymbol{w^T_{old}}\right\|$
4. Calculate X score vector $\boldsymbol{t = Xw}$
5. Calculate Y loading vector $\boldsymbol{q' = t^T Y / t^T t}$
6. Normalise Y loading vectors $\boldsymbol{q^T_{new}} = \boldsymbol{q^T_{old}} / \left\|\boldsymbol{q^T_{old}}\right\|$
7. Calculate Y score vector $\boldsymbol{u = Yq / q^T q}$
8. Repeat steps 2-7 until $\boldsymbol{t}$ converges.
9. Calculate Y loading vector $\boldsymbol{p^T = t^t X / (t^t t)}$
10. Deflate $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices ($\boldsymbol{X' = X - tp^T, Y' = Y - btc^T}$), then return to step 1 using $\boldsymbol{X'}$ and $\boldsymbol{Y'}$ for next component.

The mixed relationship can be used to predict new values of $\boldsymbol{Y}$ based on incoming $\boldsymbol{X}$ data. When predicting the output ($\boldsymbol{Y}$) data, the incoming input ($\boldsymbol{X}$) data is projected to the latent variable space by using the $\boldsymbol{X}$ loadings $\boldsymbol{P}$ obtained during the model training and using

Equation 2 (James *et al.*, 2017). Once projected the scores of the incoming data are stored as they represent the location and magnitude of the data in the latent space. These scores are used along with PLS regression coefficients and the loadings $\boldsymbol{Q}$ in Equation 10 to obtain new predictions of $\boldsymbol{Y}$ (Geladi & Kowalski, 1986).S

As the PLS regression technique is based on the PCA technique the regressed PLS retains the dimensionality reduction capabilities of the PCA technique. As the model is regressed on a dataset of reduced dimensionality this regression model can be used on datasets with many features. A possible example would be for a dataset comprising of one hundred variables the regression can be completed by projecting the dataset to ten latent variables. The regression is then performed using the projections onto the latent variables (James *et al.*, 2017).

This algorithm can be used to achieve a model which can predict outputs based on the predictor datasets used to train the model. Thus, it can be used to obtain parameter estimates for series and residuals for parallel hybrid models.

2.5.2 Dynamic PLS

A possible modification to the performance of the standard PLS model is dynamic PLS modelling. The dynamic PLS (dPLS) model introduces time lagged variables to the considered dataset as illustrated in Figure 5. In this work the time lagged variables are introduced in the predictor dataset. The introduction of time lagged variables into the PLS dataset has been done in the work conducted by Ricker (Ricker, 1988).

The aim of introducing time lagged variables into the PLS model is capture the process dynamics (Ricker, 1988). The lag variable time window width is selected to capture the process dynamic cycle. In the dPLS model the standard PLS model is trained using all entries captured within the time window. In other words, features from previous time-points are added as features for the current time-point. By capturing all states within a dynamic cycle, the PLS model could possibly predict the responses more accurately. A point to note is that by introducing lagged variables one can greatly increases the dimensionality of the predictor dataset. This in turn increases the PLS model training and execution time. Also, the introduction of lagged variables is only logical if each observation in the dataset has the same time interval between them. If the observations are not sampled at equal time intervals the model will not capture the true process dynamics for a given timespan.

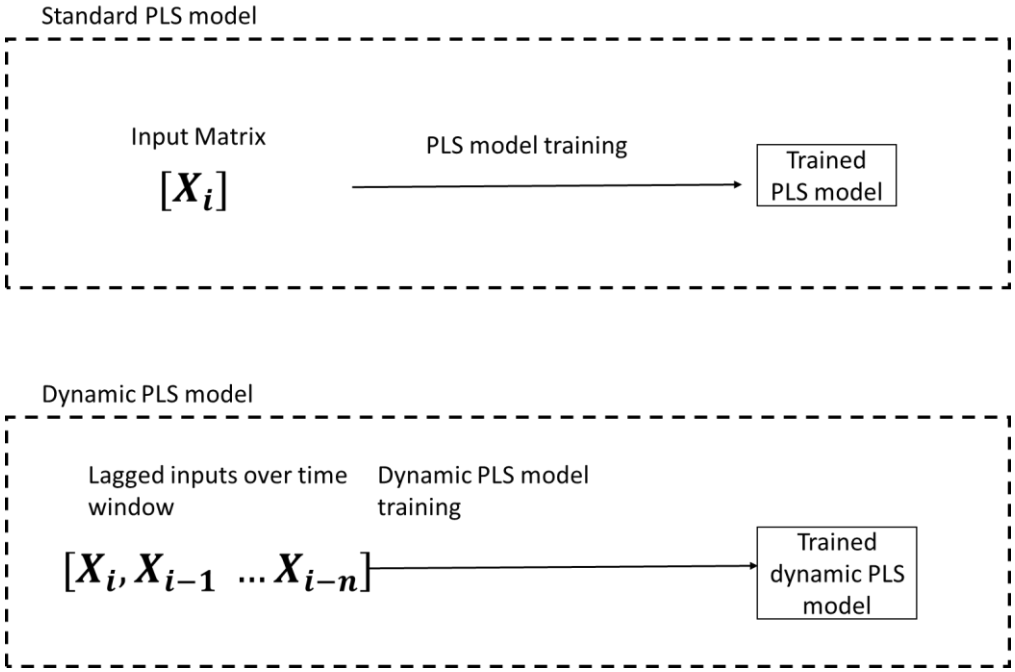

Figure 5: Illustration of time lagged input to dynamic PLS model

2.5.3 Recursive PLS model

The recursive PLS model is comprised of two standard PLS models trained using the NIPALS algorithm with one model continuously updated using information obtained from both models (Qin, 1998). The aim of the model is to capture and describe a process which undergoes time-based changes by updating the model parameters based on new data. The algorithm is shown diagrammatically in Figure 6.

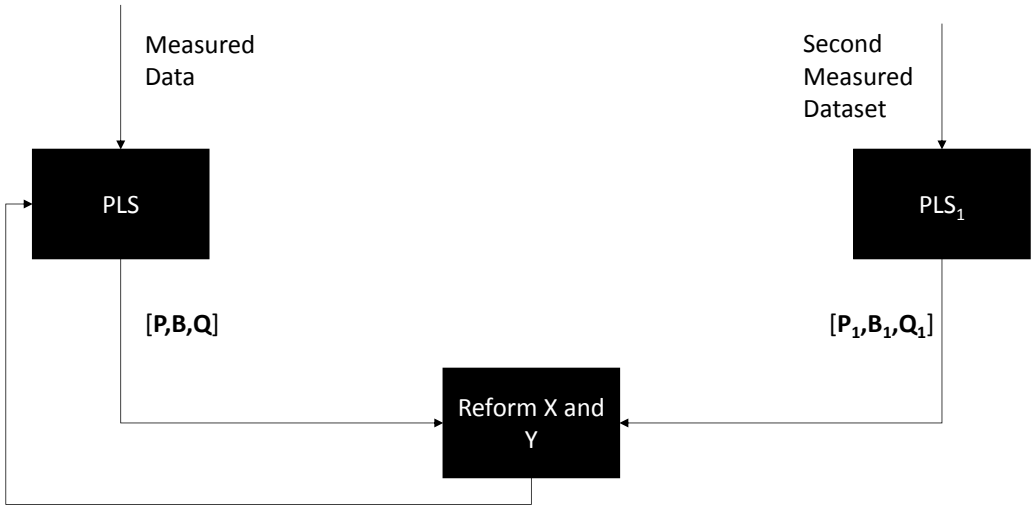

Figure 6: Recursive PLS algorithm

If one considers a time-series of process data, in the recursive PLS algorithm a PLS model is trained using available process data by the NIPALS algorithm as described in Section 2.5.1. A second PLS model is then trained using a second subsequent dataset of the same size as the first which consecutively follows the first (Qin, 1998). The PLS models are trained such that the number of PLS components are the same as the number of predictors. The $\boldsymbol{X}$ and $\boldsymbol{Y}$ loading vectors and regression coefficient vectors ($\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{B}$ respectively) are extracted from the PLS models and used to reform $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices. The reformed $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices are shown in Equations *11* and *12* below (Qin, 1998).

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{P}^T \\ \boldsymbol{P_1} \end{bmatrix} \qquad [\ 11\ ]$$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{B}\boldsymbol{Q}^T \\ \boldsymbol{B_1}\boldsymbol{Q_1} \end{bmatrix} \qquad [\ 12\ ]$$

These reformed matrices are then recursively fed back to the first PLS model and are used to retrain the model. A new set of measured data is then used to train the second PLS model and the process is repeated thus continuously updating the first PLS model (Qin, 1998). The first PLS model is thus the model used to describe the considered process or system. Once the PLS model is updated using the reformed matrices the resulting PLS model is referred to as the combined PLS model. The regression coefficients, score and loading vectors can be taken from the first PLS model to make predictions using a new set of predictors (Qin, 1998).

The recursive PLS algorithm thus becomes the sequence of steps as described below (Qin, 1998):

1. Formulate first $\boldsymbol{X}$ and $\boldsymbol{Y}$ dataset. Centre and scale the matrices
2. Derive PLS model for first dataset using NIPALS algorithm. Store $\boldsymbol{T}, \boldsymbol{U}, \boldsymbol{P}, \boldsymbol{Q}$ and $\boldsymbol{B}$.
3. When new datasets $\boldsymbol{X_1}$ and $\boldsymbol{Y_1}$ is available centre and scale the matrices.
4. Derive a PLS sub-model using the newly available data set. Store the derived $\boldsymbol{T_1}, \boldsymbol{U_1}, \boldsymbol{P_1}, \boldsymbol{Q_1}$ and $\boldsymbol{B_1}$ matrices.
5. Determine the reformed matrices $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{P}^T \\ \boldsymbol{P_1} \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{B}\boldsymbol{Q}^T \\ \boldsymbol{B_1}\boldsymbol{Q_1} \end{bmatrix}$ and return to step 2.

An adjustment to the recursive PLS method can be made through the introduction of forgetting factors. The forgetting factor acts as a weighting factor which determines the rate at which the contribution of older data to the PLS parameters decays exponentially. The reformed input and output matrices incorporating forgetting factors are shown in Equations 13 and 14 where $0 \leq \lambda \leq 1$ is the forgetting factor (Qin, 1998). A smaller forgetting factor represents a faster decay in the contribution of the older datasets (Qin, 1998).

$$\boldsymbol{X} = \begin{bmatrix} \lambda \boldsymbol{P}^T \\ \boldsymbol{P_1} \end{bmatrix} \quad [\ 13\ ]$$

$$\boldsymbol{Y} = \begin{bmatrix} \lambda \boldsymbol{B}\boldsymbol{Q}^T \\ \boldsymbol{B_1}\boldsymbol{Q_1} \end{bmatrix} \quad [\ 14\ ]$$

In order to implement the recursive PLS model with forgetting factors one follows the recursive PLS algorithm as described above. The only adjustment which occurs is replacing the reformed matrices with those given in Equations 13 and 14. By implementing the recursive PLS model in a hybrid model the aim would be to constantly update the model to account for changing process dynamics. This would in turn result in more accurate model predictions.

2.6 Support Vector Machine Classifiers

As the fault detection problem is considered a multiclass classification problem the choice of classifier is of the utmost importance. As the inputs and outputs of the model may not be linearly separable in nature a classifier which can perform well with non-linearly separable data

is required. Thus the SVM classifier is selected as it is flexible and performs well with non-linearly separable data due to the use of kernels (Vapnik, Golowich & Smola, 1997). As SVM classifiers are kernel-based methods the choice of appropriate kernel has a large impact on the effectiveness of the method as the classifier only works as well as the kernel allows it to. The kernel used for this method is the gaussian/radial basis function (RBF) kernel. The gaussian kernel is chosen as it has wide applicability and performs well with many different non-linear datasets (Bishop, 2006).

2.6.1 Linear SVM classifiers

The SVM classifier separates classes by using a hyperplane which acts as the class boundary dividing the classes (Vapnik, 1998). This is illustrated in Figure 7 below where a SVM classifier is applied to a binary class dataset. As seen in Figure 7 a linear hyperplane separates the dataset based on the data class to which each data entry belongs. The data which belongs in class 1 (shown in blue) lie on one side of the hyperplane while the data which belongs to class 2 (shown in green) lie on the other. The SVM classifies data based on which side of the hyperplane the data lie. The position of the hyperplane margin is determined by the data entries. The data entries which are in contact with the margin determine the margin width (d) and are known as the support vectors (Vapnik, 1998).

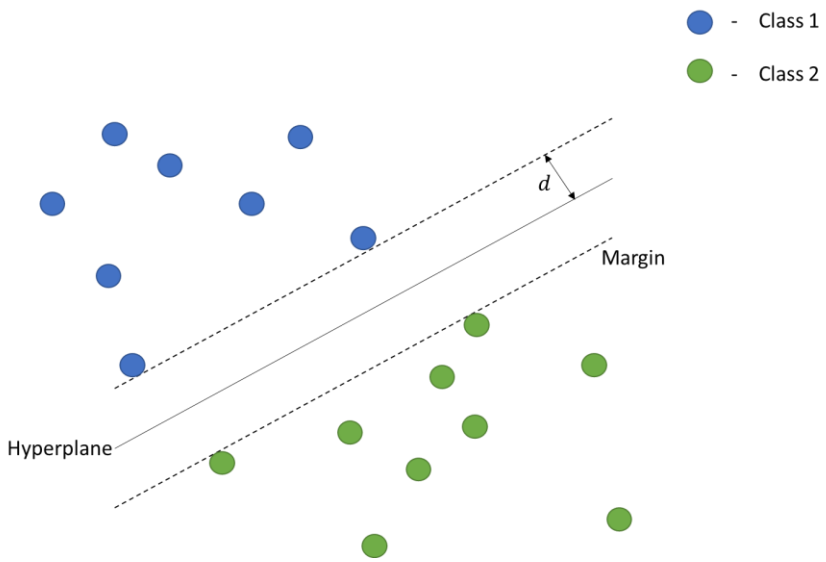

Figure 7: Illustration of a two class linear SVM classifier separating hyperplane

The hyperplane takes the form shown in Equation 15 where $\boldsymbol{w}$ represents the weight vector and b_i represents the bias. If the dataset is linearly separable then there exist a set of parameters for which the datapoints which belong to one class have predicted values (y) which are positive (the positive class) with the points belonging to the other class being negative (the negative class) (Bishop, 2006). When the class labels are considered by the SVM method the target labels (represented by l_n) for the positive class are assigned the value +1 and the labels associated with the negative class being -1 respectively.

$$\boldsymbol{w}^T\boldsymbol{x} + b_i = y(\boldsymbol{x}) = 0 \qquad [\ 15\ ]$$

The position of the hyperplane is set such that the distance between the hyperplane and the margin on either side of the hyperplane is maximised (Boser, Guyon & Vapnik, 1992). The margin surrounding the hyperplane defines the different class bounds based on the training data. The SVM method maximises the distance between the hyperplane and the margin with the position of the margins dependent on the training dataset (Christanni & Shawe-Taylor, 2000). For linearly separable datasets the training set data points do not cross the margin boundary so that all data points lie outside the margin. Thus, the maximum width of the margin at any position along the hyperplane is when the margin boundary comes into contact with the data points (Steinwart & Christmann, 2008). The use of the support vectors to define the margin width is an advantage of the SVM approach as by using support vectors the class bounds are defined by only a subset of the datapoints as opposed to all the data points. This reduces the chances of overfitting the classifier as well as improving the training speed of the classifier (Bishop, 2006).

The perpendicular distance from the hyperplane to any point $\boldsymbol{x_n}$ is illustrated in Equation 16 (Vapnik, 1998):

$$l_n y(\boldsymbol{x}_n) = l_n(\boldsymbol{w}^T\boldsymbol{x_n} + b) \qquad [\ 16\ ]$$

To ensure that only correctly labelled and identified data is considered the constraint that $l_n y(\boldsymbol{x}_n) > 0$ is introduced (i.e. if correctly classified the target label and prediction will be of the same sign thus the product will be positive) (Boser *et al.*, 1992). It is specified that the distance from the hyperplane to the nearest data points (the support vectors) is one thus the constraint

shown in Equation 17 is applied (Boser *et al.*, 1992). If the distance between the support vector and the hyperplane is one, then the distance between any datapoint and the hyperplane is equivalent or greater than one (Vapnik, 1998).

$$l_n(\boldsymbol{w}^T\boldsymbol{x_n} + b) \geq 1 \qquad [\ 17\ ]$$

The objective of the SVM method is to maximise the margin width. This can be expressed as finding the weights which maximise the boundary width (i.e. Equation 16) (Vapnik *et al.*, 1997). Smaller weights will result in an invalid margin boundary in which data points will lie between the margin boundary and hyperplane. Thus, the solution to the SVM becomes finding the minimum weights for which the constraint in Equation 17 holds. This is represented as the optimization problem shown in Equation 18 subject to the constraints shown in Equation 19 (Hastie, Tibshirani & Friedman, 2001).

$$\min_{\boldsymbol{w},b} \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 \qquad [\ 18\ ]$$

Subject to:

$$l_n(\boldsymbol{w}^T\boldsymbol{x_n} + b) \geq 1 \qquad [\ 19\ ]$$

In order to present a more simplified form of the optimization problem Lagrange multipliers are applied to the problem and simplified to present the dual representation form as shown in Equation 20 where N represents the number of samples in the training set and $\boldsymbol{a}$ the Lagrange constants (Boser *et al.*, 1992).

$$\max_{a} \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m l_n l_m \boldsymbol{x}_n^T \boldsymbol{x}_m \qquad [\ 20\ ]$$

Subject to the constraints:

$$a_n \geq 0,\ \sum_{n=1}^{N} a_n l_n = 0 \qquad [\ 21\ ]$$

The above solution describes the hard margin approach to SVM classifiers in which the dataset must be clearly linearly separable. This is not always the case as often classes tend to have slight overlaps or class bounds that cannot linearly separate every datapoint. To the address

these issues the regularized approach to SVM classifiers (also known as the soft-margin approach) is adopted. Slack variables are introduced to loosen the restrictions on all datapoints having to be outside the margin boundaries(Bishop, 2006). This allows some data entries to cross the class boundaries. The regularised Lagrangian form of the optimization equation is shown in Equation 22 below with the constraints given in Equation 23 (Vapnik, 1998). As seen the objective function remains the same with the only change being to the constraints with the addition of the regularization constant (C) which accounts for the tolerance of allowing data points behind the margin boundary.

$$\max_{a} \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m l_n l_m \boldsymbol{x}_n^T \boldsymbol{x}_m \qquad [\ 22\ ]$$

Subject to the constraints:

$$C \geq a_n \geq 0,\ \sum_{n=1}^{N} a_n l_n = 0 \qquad [\ 23\ ]$$

2.6.2 Kernel SVM classifiers

The above discussion considers datasets which are linearly separable and will not perform well on a training dataset which cannot be separated linearly. This issue is addressed through the use of kernels transformations. Kernels are used to transform the dataset to a space in which the data can be linearly separated and the SVM applied (Christanni & Shawe-Taylor, 2000).

Kernel transformations are used to map a non-linear dataset to some mathematical space (referred to as the kernel space) where the data becomes linear in nature. One can consider a dataset $\boldsymbol{X}$ which is not linearly separable. The dataset can be mapped to a higher dimensional feature space through a mapping function Φ (Bishop, 2006). In the new feature-space the data could possibly become linearly separable. The mapping of a dataset to a new feature space is shown in Figure 8. As seen the data is transformed from a two-dimensional space to higher dimensional, three-dimensional space where the data becomes linearly separable.

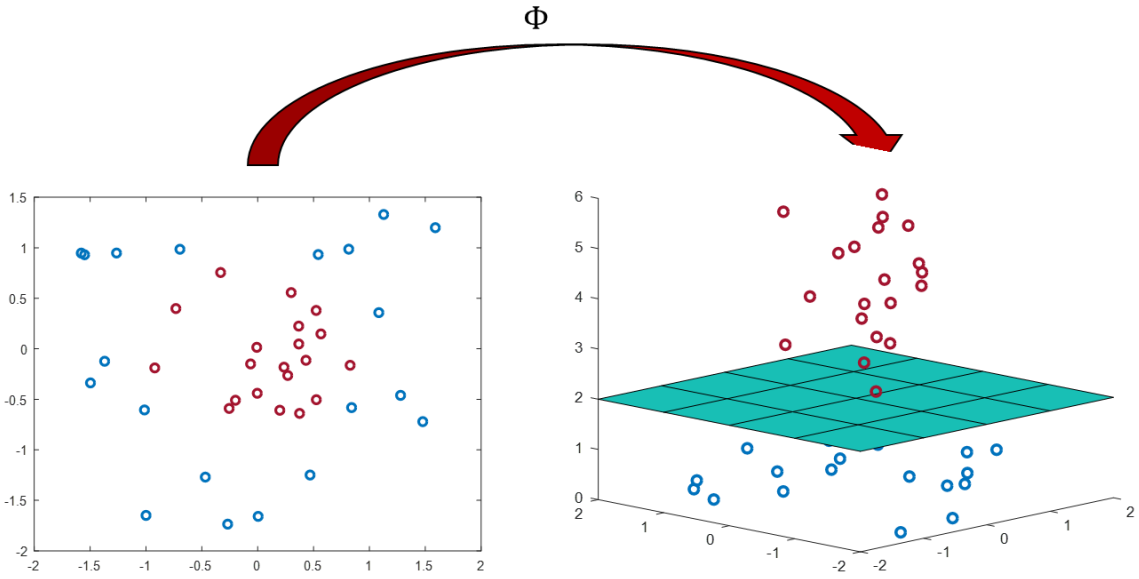

Figure 8: Mapping of non-linearly separable data

The mapped dataset is represented by $\Phi(x)$. Mapping each observation using a transformation function becomes a time consuming and computationally intensive process as it requires one to determine the function which results in the data becoming linear. This gives rise to what is known as the 'kernel trick' in which the product of two mapped data vectors is replaced by a kernel function (Aizerman, Braverman & Rozonoer, 1964)(Steinwart & Christmann, 2008). This results in only the pairwise comparison of the observations being considered thus removing the need to explicitly map each individual observation (Aizerman *et al.*, 1964). Thus, a kernel function can be defined as shown in Equation 24 in which the kernel function is equivalent to the dot product of two mapped data vectors (Steinwart & Christmann, 2008).

$$k(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) \qquad [\ 24\ ]$$

Kernel functions have characteristics which define them with one of those being that the function must be positive definite (Steinwart & Christmann, 2008). The kernel function is then used to generate the Gram matrix which contains the results of the kernel function applied to all combinations of the input data(Bishop, 2006). The Gram matrix is shown in Equation 25.

$$\boldsymbol{K} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_n, x_1) \\ \vdots & \ddots & \vdots \\ k(x_1, x_n) & \cdots & k(x_n, x_n) \end{bmatrix} \qquad [\ 25\ ]$$

When a SVM classifier is applied to a mapped dataset the position of the hyperplane is determined in the mapped feature space. The Lagrange form of the optimization objective function with mapped data is shown in Equation 26 (Christanni & Shawe-Taylor, 2000).

$$\max_{a} \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m l_n l_m \Phi(\boldsymbol{x}_n)^{\mathrm{T}} \Phi(\boldsymbol{x}_m) \quad [\ 26\ ]$$

As seen the optimization problem consists of the product of the mapped data as an input. The kernel trick can be applied to replace the product of the mapped data vectors with a kernel function. The kernel SVM optimization problem is now defined as seen in Equation 27 with the kernel function inserted (Boser *et al.*, 1992). The performance of the kernel SVM method is heavily dependent on the selection of an appropriate kernel which transforms the data to some space in which the data can be separated by the SVM method.

$$\max_{a} \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) \quad [\ 27\ ]$$

Subject to the constraints:

$$C \geq a_n \geq 0,\ \sum_{n=1}^{N} a_n t_n = 0 \quad [\ 28\ ]$$

2.6.3 Kernel functions

As non-linear SVM implements kernel transformations for non-linear mapping the choice of appropriate kernel function is key. There are several kernel functions available with a few examples being linear kernels, polynomial kernels, sigmoidal kernels and radial basis function (RBF) kernels. The choice of kernel depends on the dataset considered.

The RBF or Gaussian kernel is a commonly used kernel function in SVM (Bishop, 2006). This is due to the kernel performing well on several different non-linear datasets. The RBF kernel function used in this work is shown in Equation 29. The width parameter (γ) is optimized to ensure the best fit for the RBF kernel.

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-\gamma \left\| \boldsymbol{x_i} - \boldsymbol{x_j} \right\|^2\right) \quad [\ 29\ ]$$

The polynomial kernel is used when the feature space can best be described using a polynomial function. The polynomial kernel is shown in Equation 30 below (Yekkehkhany, Safari, Homayouni & Hasanlou, 2014). The degree of the polynomial is set through the parameter p.

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(\boldsymbol{x_i^T x_j} + 1\right)^p \quad [\ 30\ ]$$

The sigmoid kernel is given in Equation 31 below.

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tanh\left(\boldsymbol{x_i^T x_j} + 1\right) \quad [\ 31\ ]$$

Due to the generalisation capabilities of the RBF kernel, it is often the first kernel function applied in models which use kernel transformations (Bishop, 2006). If the RBF does not perform well or if some knowledge is known about some underlying model which describes the data, a kernel which better fits the data can be selected. The selection of an appropriate kernel function can become a trial-and-error approach. In this work the RBF kernel is selected due to its generalisation ability.

2.6.4 Multiclass SVM classifiers

SVM classifiers operate as binary classification models thus to be applicable to fault identification an adjustment is needed to allow the SVM to effectively handle multiclass classification. The one-vs-one multiclass SVM approach is adopted to solve the proposed multiclass classification problem (James *et al.*, 2017). As traditional SVM classifiers are designed for two class classification the one-vs-one approach involves using SVM classifiers being trained pairwise until all class combinations are accounted for. The pairwise classifiers for three classes are represented schematically Figure 9 in below.

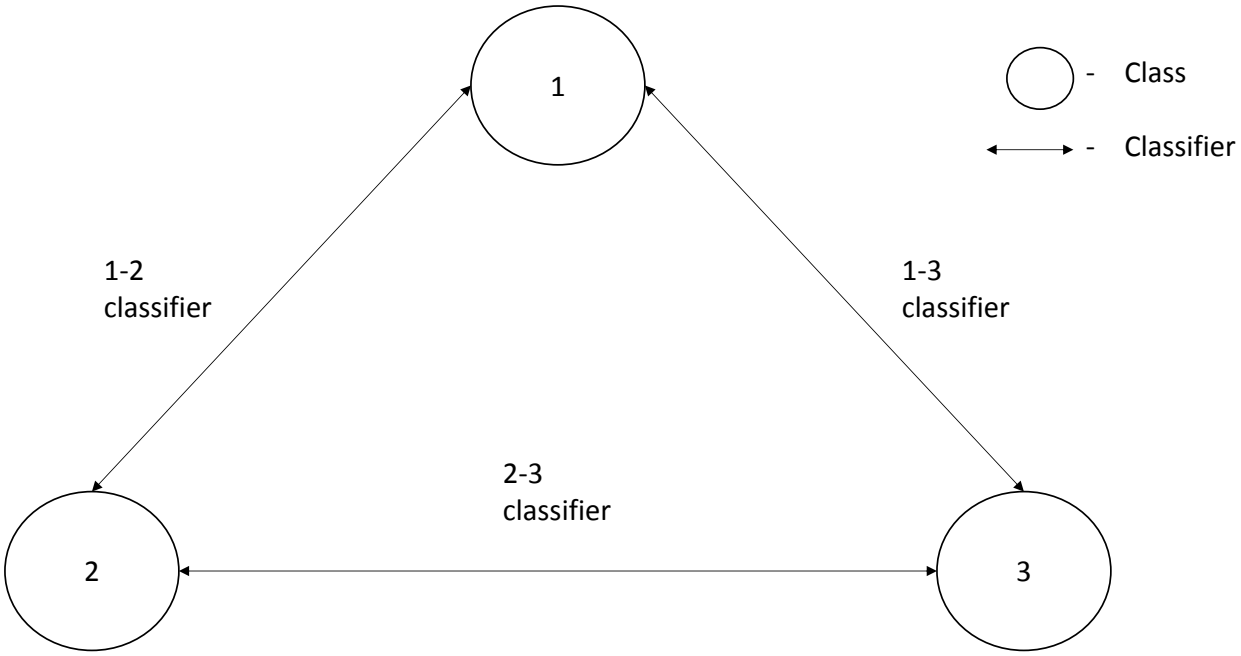

Figure 9: Pairwise SVM classifiers for three classes

When classifying new data points the data points are fed to all the classifiers. The resulting output labels from each classifier for each data point are collected and the number of labels corresponding to each fault for each observation is recorded. The label with the largest number of positive identifications is assigned as the most likely label for that observation. In this work the pairwise classifiers are trained using normal operation data as well as the fault laden datasets. The fitcsvm function in MATLAB is used to train all the SVM classifiers. Two-class SVM's are trained on labelled process datasets. The classification predictions using the trained SVM classifiers are performed in MATLAB using the predict function. The one-vs-one approach is then applied to the prediction of new classes with the predicted labels summed to determine the most likely label.

2.7 Cross Validation

Cross validation is a data partitioning and error estimation method in which a dataset is subdivided into separate groupings in order to evaluate the performance of various statistical methods and machine learning algorithms (Stone, 1974). The data is portioned into two groups namely a training set which is used to train the machine learning technique and a test set which is used to evaluate the performance (Stone, 1974). The cross validation of datasets is particularly useful when the available process or system data is limited. By cross validating the

dataset one can train a model using the available data multiple times using different groupings within the dataset while still being able to determine the performance of the model using the test set.

There are several cross-validation techniques available with an example technique being hold-out cross validation. Hold-out cross validation comprises of partitioning a portion of the data (usually represented as a percentage) as the test set with the rest allocated as the training set (Kohavi, 1995). The holdout method has one noticeable and distinct disadvantage when compared to other cross validation methods in that it involves sub dividing the data set only once. The data which falls into the test set is thus never used to train the model and the performance of the model is only considered once.

Another cross-validation technique is k-fold cross validation. This technique comprises of subdividing the data into several partitions (determined by the k factor) of equal size (Bishop, 2006). One of these partitions is allocated as the test set with the rest being the training set. The model is then trained and tested, and the process is repeated by selecting the next partition as the test set and allocating the rest as the training set. The loop is repeated until all partitions have been selected as the test set. The k-fold cross validation ensures that the machine learning technique has be trained on all available data. This ensures that one gains the most use out of a limited dataset.

2.8 Performance measures

When evaluating the performance of the fault detection method there are different measures which can be used to quantify the performance and fault identification capabilities of the classifier. A performance measure used to quantify the performance of the fault identification method is the sensitivity measure. The sensitivity is evaluated according to Equation 32 where true positive represents the fault datapoints correctly identified as fault points and false negative represents fault datapoints misclassified as normal operation datapoints (non-fault datapoints) (James *et al.*, 2017). The sensitivity represents the likelihood that fault datapoints are correctly classified as such.

$$sensitivity = \frac{true\ postive}{true\ positive + false\ negative} \quad [\ 32\ ]$$

Another commonly used fault identification performance measure is the specificity. The specificity is defined in Equation 33 where false positive represents the number of datapoints misclassified as positive fault readings (James *et al.*, 2017). True negative represents the normal operation data which is correctly identified and false positive represents normal operation data which is incorrectly identified as faulty. The specificity represents the likelihood that a datapoint identified as a non-fault point is in fact a non-fault point and not a false identification.

$$specificity = \frac{true\ negative}{true\ negative + false\ positive} \quad [\ 33\ ]$$

The sensitivity and specificity of the fault identification method are important measures and as such are commonly displayed simultaneously as receiver operating characteristic (ROC) curves (James *et al.*, 2017). ROC curves are commonly used to compare the performances of different classifiers as the ROC displays both significant performance measures simultaneously.

Many classification methods make use posterior probability thresholds with the thresholds determining which class a datapoint will be classified as. The ROC then displays the values of sensitivity and 1-specificity for all values of the probability simultaneously (James *et al.*, 2017). The overall performance of the classifier can then be quantified by using the area under the curve (AUC).

The SVM method however does not make use of probability thresholds with classification of data points being solely dependent on which side of the hyperplane that datapoint lies on. Posterior probabilities of the SVM can be estimated however by fitting a posterior probability curve to the SVM score vectors. There are different methods in which this can be done with one such method is Platt scaling which is implemented by using the fitPosterior function in MATLAB (Platt, 2000).

2.9 Platt scaling

A noticeable feature of SVM classifiers is that the position of the support vectors (and as such the decision boundaries) are determined by only maximising an objective function (Boser *et al.*, 1992). The output of the SVM is therefore only the class to which observations belong with no information on the likelihood or probability that observations belong to the resultant class.

Probabilistic outputs for classifiers provide useful information which can be used to further investigate and evaluate the classifier performance and make more informed classification decisions (Platt, 2000). One method of obtaining probabilistic outputs for SVM classifiers is Platt scaling (Platt, 2000).

Platt scaling is a post-training method of obtaining probability outputs for trained SVM models by fitting a parametric sigmoid function to the output of the SVM classifier (Platt, 2000). The parametric sigmoid function is used to fit the posterior probability $P(l = 1|f)$ (i.e. probability that an observation lies on one side of the hyperplane) directly. Platt examined empirical data and based on the data stated that the class-conditional probabilities between the hyperplane margins are exponential (Platt, 2000). The sigmoid function form is determined by Bayes' rule on two exponentials. The parametric sigmoidal function takes the form shown in Equation 34 where f is the SVM output.

$$P(l = 1|f) = \frac{1}{1+\exp(Af+B)} \quad [\ 34\ ]$$

The sigmoid parameters (A and B) are determined by minimizing a negative likelihood function. In order to generate the likelihood function, a method of mapping the SVM outputs to representative probabilities must be set. This is done by setting a new training set (f, p_t) where p_t is a target probability as defined in Equation 35 (Platt, 2000).

$$p_t = \frac{l+1}{2} \quad [\ 35\ ]$$

The likelihood function (given in Equation 36 below) is minimized by adjusting the sigmoid parameters to ensure the best fit of the probability mapping function. The minimization can be performed by using an appropriate optimization routine.

$$\min\left(\sum_i p_{t_i}\log(p_i) + \left(1 - p_{t_i}\right)\log(1 - p_i)\right) \quad [\ 36\ ]$$

Where:

$$p_i = \frac{1}{1+\exp(Af_i+B)} \quad [\ 37\ ]$$

Platt scaling is advantageous as it obtains probabilistic outputs using the outputs of a trained SVM model. Other methods can used to determine probabilities for SVM models such as the method proposed by Wahba (Wahba, 1992). The method involves introducing a maximum likelihood function during the training of the SVM function (Wahba, 1992). This could affect the position of the support vectors and thus the classification efficiency of the SVM model. Platt scaling however bypasses these concerns by being a post-training probability mapping method

3 LITERATURE REVIEW

The following chapter reviews applications of hybrid modelling, fault detection and fault identification in available literature. The applications of the method in literature are described and the results of the studies conducted are summarised. The application of both hybrid modelling and the fault detection and identification are evaluated and compared.

3.1 Fault identification using k-nearest neighbours

There has been much work done in the application of data-based classification models in fault detection and identification. These approaches pose fault identification as a classification problem with each class corresponding to a different fault condition. This was the approach followed by He and Wang (2007) who made use of k-nearest neighbour (kNN) classification models .

He and Wang (2007) considered the use of the kNN classification model as a fault detection method. The kNN was trained on labelled datasets which contained both normal operation as well as faulty operation data. The performance of the kNN fault detection was compared to a PCA method utilising squared predictive error (SPE) and a PCA method utilising Hotelling's T^2 statistic (He & Wang, 2007).

The kNN fault detection methodology proposed by He and Wang (2007) comprises of two parts, namely a model building part and a fault detection part. In the model building component for each observation in the training set the k nearest neighbours were identified and the distances were calculated. Once the distances were determined the fault detection threshold was set (He & Wang, 2007). In the fault detection component when an unidentified observation was subjected to the fault detection the kNN for the observation among the training set were identified (He & Wang, 2007). The distance between the observation and its neighbours were calculated and compared to the fault detection threshold. If the distance exceeded the threshold value that observation was identified as being faulty (He & Wang, 2007).

It is noted that the performance of the kNN model can decrease when applied to a dataset comprising of a large number of input variables (He & Wang, 2007). This is due to the model treating each variable as equally important; thus it can classify data using irrelevant features.

By introducing some data pre-processing and feature extraction or feature selection the performance of the model can be increased (He & Wang, 2007). In this work however no feature extraction was used as the feature extraction requires human interaction and the aim was to automate the fault detection method as much as possible (He & Wang, 2007).

The fault detection methods were applied to a process dataset obtained from an industrial process and the performances were compared. The industrial dataset contained twenty different process faults with the performance of the fault detection methods evaluated on each process fault. When the performances were compared it was seen that the PCA model using SPE was only able to correctly identify a small subset of the process faults with 11 out of the possible 20 identified (He & Wang, 2007). The PCA model using T^2 was only able to correctly identify 10 out of the possible 20 process faults. The kNN model however showed an increase in performance correctly identifying 17 of the possible 20 process faults (He & Wang, 2007). This demonstrates that the kNN model can be used a viable fault identification method (He & Wang, 2007).

3.2 Fault identification using artificial neural networks

As seen the classification problem approach to fault identification is viable approach. Similar approaches were followed using different classification model types with each yielding a different result based on the model used. Heo and Lee (2018) followed a similar approach utilising artificial neural networks (ANN).

Heo and Lee (2018) implemented an ANN based fault detection and fault classification model. The fault detection and classification were treated as a classification problem and ANN's were developed to address the classification. The fault identification model was applied to the Tennessee Eastman process.

For the fault detection method binary ANN's were trained on normal operation data and faulty operation data (Heo & Lee, 2018). Binary ANN's were then trained on each process fault in the process. The number of neurons in the ANN's were adjusted and the fault detection performance was recorded. It was seen that when the ANN contained two hidden layers the fault detection performance increased greatly when compared to an ANN which only contained one hidden layer for all faults considered (Heo & Lee, 2018). Increasing the number of hidden layers beyond two yielded no significant increase in performance (Heo & Lee, 2018).

A data augmentation approach was attempted in which consecutive observations were combined and evaluated simultaneously by the ANN's. When this approach was followed it was found that a significant increase in the detection accuracy occurred (Heo & Lee, 2018). The ANN fault detection method was compared to fault detection models using dynamic PCA, modified PCA, independent component analysis and deep belief networks using both Gaussian and sigmoid activation functions. It was seen that the ANN fault detection method on average showed a higher detection rate than the other fault detection models as well as a significantly lower false alarm rate (Heo & Lee, 2018).

The fault classification was posed as a multiclass classification problem where normal operation and each fault was designated as its own class (Heo & Lee, 2018). Through this approach, both fault detection and classification occurred simultaneously (Heo & Lee, 2018). The performance of the fault classification method was compared when the standard dataset and augmented dataset are used. It was found that by performing the data augmentation the performance of the fault identification method increased (Heo & Lee, 2018).

The fault identification performance was then compared to fault identification methods which used shallow neural networks, hierarchical neural networks and a stacked sparse autoencoder. It should be noted that the considered models only perform fault identification and thus are only trained on fault data and do not include normal operation as a class (Heo & Lee, 2018). It was seen that the ANN model showed significantly higher fault classification rates than the other considered fault identification models (Heo & Lee, 2018). This not only shows that ANNs are a viable fault identification model but also shows that the multiclass approach to fault identification is valid. Training a simultaneous fault detection and identification model is also an approach which performs well.

3.3 Fault identification using support vector machines

As seen not only does the classification approach to fault identification work with other classification models but a dual fault detection and identification approach is a viable approach as well. Another possible classification model which could be used in fault detection and identification is the SVM classifier. The SVM classifier has shown good classification performance for both linearly and non-linearly separable datasets thus could possibly work well

as fault identification model. One example of SVM fault detection was done by Yin *et al.* (2014).

A support vector machine classifier was trained as a fault detection model and tested using the Tennessee Eastman process (Yin *et al.*, 2014). The Tennessee Eastman process model is a process simulation model widely used in process control. This is due to wide array of process faults which can be applied to the model thus making it a suitable test model for fault detection techniques (Yin *et al.*, 2014). The performance of the SVM fault detection model was compared to a PLS fault detection model.

As the standard SVM is a binary classification model, for fault detection the SVM was trained using normal operation data as one class and data which corresponds to when a process fault was active as the other. The SVM was proposed as the fault detection method due to it generalization capabilities as well as its ability to classify non-linearly separable data due to the use of kernel transformations. The SVM classifier was trained and evaluated on one fault condition at a time thus following a binary classification approach and not a multiclass approach (Yin *et al.*, 2014). The radial basis function was selected as the kernel used by the SVM. The PLS fault detection model was a standard PLS regression model trained using the normal operation data. Hotelling's T^2 statistic was used to detect when the process deviates from standard operation. When the T^2 exceeded a threshold calculated based on confidence intervals, the process was then deemed to be under faulty conditions (Yin *et al.*, 2014).

When the performance of the SVM fault detection model was compared to the PLS fault detection model it was seen that the SVM consistently performed better than the PLS model across all process faults considered. This shows that not only is a SVM fault detection model a viable fault detection method but the ability of the SVM classifier to effectively separate non-linearly separable data transfers well to fault detection (Yin *et al.*, 2014). This is due to many process datasets containing data which are non-linear in nature. The performance of the classifier with non-optimized hyperparameters was also considered and it was found that optimization of the hyperparameters greatly improved the predictive performance of the fault detection (Yin *et al.*, 2014).

The performance of the SVM model was also tested when fault laden classes were created. The fault laden class was the result of a combination of fault condition datasets with the largest combination being of three separate fault conditions. A test data set comprising of a selection of

data generated under one fault condition was then fed to SVM classifier. It was found that under these conditions the SVM fault detection model still performed well (Yin *et al.*, 2014). This shows that the SVM classifier is a viable fault detection method (Yin *et al.*, 2014). In all conditions considered in this work however the SVM classifiers were binary classifiers. This would not be ideal for fault identification models as the classifier would possibly have to distinguish between several different faults. This would result in a multiclass classification problem and as the SVM is a binary classifier and adaption is needed to account for the additional classes.

As seen the SVM model can be used as fault detection technique however it's capability as a fault identification model still needs to be evaluated. This was done in the approach followed by Onel et al. (2019). Not only is SVM as fault identifiers evaluated but also if the introduction of selected features improves the performance of the SVM classification model.

In the work support vector machines were trained to perform feature selection, the selected features were then fed to SVM classifiers used for fault detection with the intention that the selected features would improve the fault detection performance (Onel *et al.*, 2019). The fault detection technique was then applied and tested using the Tennessee Eastman process. The performance of the feature selection fault detection model was compared to the standard SVM fault detection model.

In order to perform the feature selection a binary selection variable $z \epsilon \{0,1\}$ was introduced to select or discount a feature f. Each value of the selection variable corresponds to whether a feature is selected ($z_f = 1$), or whether the considered feature is discounted ($z_f = 0$). The SVM optimization problem with the selection variable included is shown in Equation 38 (Onel *et al.*, 2019).

$$\min_{z} \max_{a} \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n \cdot z_n, x_m \cdot z_m) \quad [38]$$

Subject to constraints:

$$\sum_{n=1}^{N} a_n y_n = 0 \, , a_n \epsilon [0, C] \quad [39]$$

$$\sum_{f} z_f = m \, , z_f \epsilon \{0,1\} \quad [40]$$

Where m represents the optimal size of the reduced feature space. Solving the optimization problem to obtain a global solution in real world applications would be time consuming and impractical. A sensitivity analysis was performed on the inner optimization instead with respect to z_f (Onel *et al.*, 2019). In order to derive the sensitivity of the inner optimization the partial derivative of the transformed Lagrange function was taken. From the resultant function the criterion function is obtained and is given in Equation 41 (Onel *et al.*, 2019).

$$crit_f = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n^* a_m^* t_n t_m \frac{\delta(k(x_n \cdot z_n, x_m \cdot z_m))}{\delta z_f}\Big|_{z=z^*} \quad [\ 41\ ]$$

$$f_{worst} = \arg\max_f crit_f \quad [\ 42\ ]$$

In order to perform the feature selection an iterative procedure was followed by which SVM models were trained. The features used in the training were then ranked according to the criterion value obtained in Equation 41. The feature with the lowest criteria value was then eliminated and the iterative loop repeated excluding the eliminated feature (Onel *et al.*, 2019). The loop was then repeated until a predetermined feature space size is obtained. The feature selection was then performed for each error case.

Similar to the work performed by Yin et al. (Yin *et al.*, 2014), SVM classifiers were trained to detect each individual fault with the classifiers trained with normal operating data forming one class and fault laden data the other. When the feature selection fault detection method was applied only the selected features are used to train the SVM classifiers. It was seen that when the selected features were used for the fault detection there was noticeable improvement in the accuracy and decrease in the false detection rate. This shows that by including a feature selection (or a possible feature extraction) method with a fault detection or diagnosis method, the performance of said method is improved (Onel *et al.*, 2019). By including a feature extraction method with a fault diagnosis method, the increase in performance could be worth the additional training and execution cost incurred by the additional model.

3.4 Model-based fault identification using multiparametric programming

Introducing selected features to fault identification models was shown to increase the performance of the models thus the introduction of relevant information to the classifier increases fault identification performance. There are different methods of obtaining relevant information from a dataset with one possible way being based on ideas found in model-based fault identification. Estimating model parameters for fault detection was done in the work by Che Mid and Dua (2017).

A parameter estimation method using multiparametric programming has been proposed as a possible fault detection method (Che Mid & Dua, 2017). Multiparametric programming was used to develop a function which determines the model parameters as a function of the measured process data. The proposed method was tested on a single stage evaporator and a quadruple tank system case study (Che Mid & Dua, 2017).

In the proposed method the parameter estimates were obtained using multiparametric programming. In parameter estimation fault detection, the parameters are obtained from non-linear ordinary differential equations by minimizing an error measure by adjusting the parameter values (Che Mid & Dua, 2017). The parameter estimation can be performed with the differential equations by using different regression techniques.

In the proposed multiparametric programming approach the first step in obtaining the model parameters is reforming the differential equations as a non-linear program by discretising the differential equations. The Karush-Kuhn-Tucker conditions for the non-linear program are obtained and the equality constraints are solved (Che Mid & Dua, 2017). By solving the equality constraints, one obtains a set of parametric algebraic equations which can be used to obtain model parameters as a function of process measurements. When performing the fault detection, the residuals between the estimated model parameters and the model parameters at normal operation are calculated. When the residuals exceed some threshold value a fault is detected, and an alarm is raised (Che Mid & Dua, 2017).

When the fault detection method was applied to the single stage evaporator system case study it was seen that the fault detection method was able to correctly identify different faults applied to the model (Che Mid & Dua, 2017). The same was found when the technique was applied to

the tank system case study. This shows that the incorporation of multiparametric modelling in the parameter estimation is a worthwhile endeavour. By including the multiparameter models in the fault detection, the model based parameter estimation fault detection can be implemented online (Che Mid & Dua, 2017). By replacing the differential equation parameter estimation with simple algebraic relationships, the computational costs are significantly reduced.

3.5 Hybrid modelling for model parameter estimation

Estimating model parameters for fault identification and including these parameters in the fault identification model was a possible method of improving fault identification performance. When these parameters were included in the fault identification model an increase in performance was noted. A possible parameter estimation method is hybrid modelling. As hybrid models combines first-principle modelling with data-based modelling it could be a viable method for parameter estimation. One example of hybrid modelling is found in the work done by Hu *et al.* (2011).

In the work done by Hu et al. an acid leaching process was modelled using hybrid modelling techniques (Hu *et al.*, 2011). The hybrid model proposed in the work made use of a bagging support vector regression (SVR) method which incorporated negative correlation learning. The performance of the proposed hybrid model was then compared to hybrid models which incorporated regular SVR, bagging and a model comprised of the mechanistic model only.

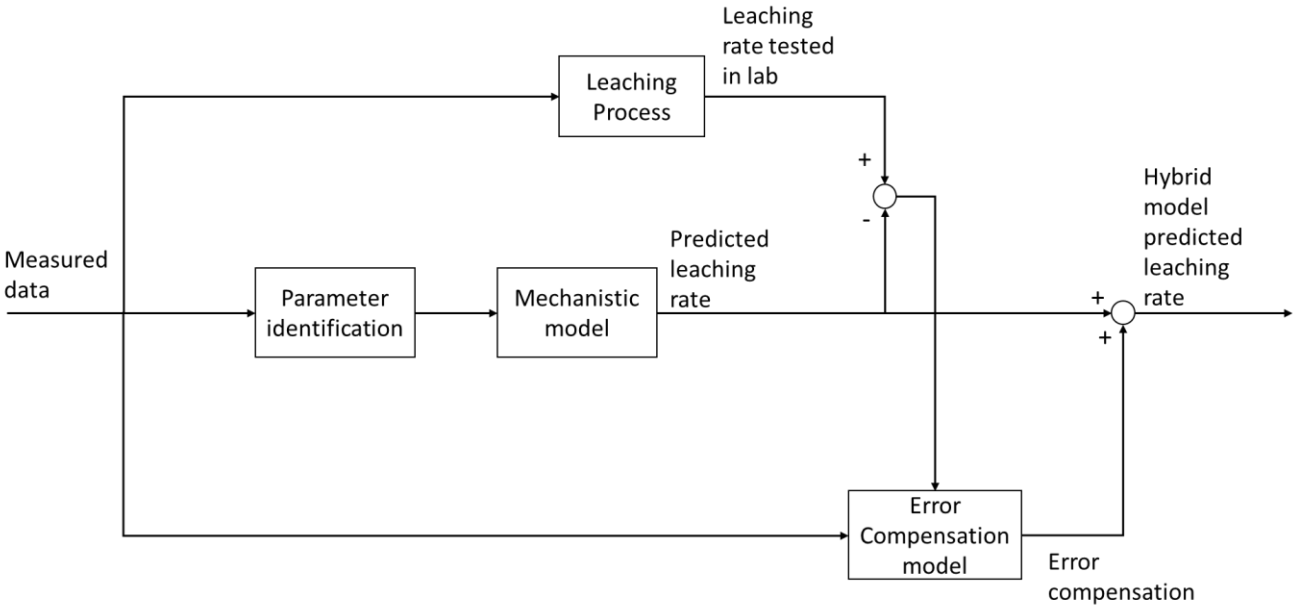

***Figure 10: Hybrid series and parallel combination model structure as adapted from Hu et al* (Hu *et al.*, 2011)**

The model structure proposed in the paper comprised of a combination of series and parallel hybrid model structures. The series component is responsible for estimating the parameters of the mechanistic model and parallel component is responsible for estimating and compensating for the fault. The hybrid model structure as adapted from Hu et al. is shown in

Figure 10 above (Hu *et al.*, 2011).

The series and parallel combination hybrid model presents a model structure which could include the advantages of both series and parallel structures thus providing a better performing model. A disdavatage of the combination structure is that by employing two emperical models the training and execution time of the hybrid model increases. Thus the use of the combination model is only justified if the model signifcantly improves on the performance of the model when using classical hybrid model structures. Another possible disadvantage is that by introducing an addition empirical model the risk of over-fitting increases as it introduces additional parameters within each empirical model.

The negative correlation SVR bagging algorithm employed in the paper was introduced as an improvement on the standard SVR algorithm which addressed the issue of over-fitting. The bagging component of the algorithm reduced the bias incurred during the training stage by drawing random samples from the training dataset. This improved the generalisation

capabilities of the algorithm by reducing the generalisation error (Hu *et al.*, 2011). The negative correlation learning component of the algorithm altered the training procedure of the SVR by introducing a correlation penalty. The algorithm comprised of training SVR's using the negative correlation method and then combining the SVR's obtained by this method. The full algorithm can be found in the paper written by Hu et al (Hu *et al.*, 2011).

The performance of the negative correlation SVR bagged hybrid model (NSBH) was compared to the performance of the mechanistic model (MM), basic bagging model (BBH) and SVR model. It was found that the NSBH model had better performance and lower error evaluations than the other models (Hu *et al.*, 2011). Due to this the NSBH algorithm showed potential as a possible hybrid modelling approach. The NSBH approach could help alleviate the over-fitting problem which is prevalent in hybrid modelling (Hu *et al.*, 2011). However, if the regular SVR is sufficient then the NSBH model will most likely not be applied as it is a more complex formulation which could possibly lead to errors in implementation and adjusting to fit other models.

Another form of hybrid modelling was found the work done by Bhutani *et al.* (2006). In the work performed by Bhutani *et al.* (2006), an industrial petrochemical hydrocracking unit was modelled using hybrid modelling techniques. The hybrid modelling incorporated the use of artificial neural networks (ANN's) as the empirical model in differing hybrid model structures namely series, parallel and series-parallel combination model structures. The performance of the differing model structures was then compared to each other as well as a purely mechanistic model and a purely data-based/empirical model. Lastly the performance of the hydrocracking unit was optimised using the data-based model.

In the series hybrid model, the ANN was used to predict the kinetic parameters of the hydrocracking unit while in the parallel configuration the ANN was responsible for predicting the residuals. Although the series-parallel combination model used ANN's to perform the afore mentioned functions the ANN in the parallel component of the combination model and the parallel structure were different while the series component ANN was the same. This resulted in different training times and performances for the different hybrid models with the combination model taking longer to train (Bhutani *et al.*, 2006). This further reiterates the necessity of only using the more complex model structure if the performance is significantly improved.

Once the respective models were trained the performance of each model was compared. The performance of each model was compared by determining the average absolute percent error obtained by comparing the output predictions of each model to plant data. It was seen that the series-parallel performance did noticeably improve on the performance of the series structure but did not significantly improve on the performance of the parallel structure (Bhutani *et al.*, 2006). Thus, one would select the parallel structure due to the lower training time. It was found that the ANN with no hybrid modelling performed the best overall and this coupled with the short training time for the model lead the authors of the paper to optimise the hydrocracking unit based on this model. This further indicates that the selection of hybrid model structures and the use of data-based models vary on a case-to-case basis.

The use of ANN in hybrid modelling was seen in the work of Conlin *et al.* (1997) and Lee *et al.* (2002). In the work done by Conlin *et al.* (1997) hybrid modelling techniques were used to predict the pressure drop in a water treatment plant. The pressure loss profiles in the water treatment plant were modelled using both parallel and series hybrid structures. The ability of the hybrid models to correctly predict the pressure loss were then compared to each other as well as the first-principle model.

The first-principle model considered was a polynomial relationship which describes the head loss pressure profile. The data-based model considered in the work was an ANN. In the series structure the ANN was used to predict parameters used by the mechanistic model (Conlin *et al.*, 1997). The parameter prediction can be viewed as a feature extraction method as it extracts the parameters by predicting them using the input data. In the parallel structure the ANN was used to estimate the output residuals based on the input data.

When the performance of the parallel hybrid model was compared to that of the standard mechanistic model it was seen that both models accurately described the pressure loss which occurred. However, when a step input was applied to the system it was seen that the mechanistic model no longer accurately described the pressure. It was also seen that, although showing a slight improvement over the mechanistic model, the parallel structure did not account for the vast change in pressure drop due to the step input. This could be attributed to the parallel structure not being able to extrapolate the residuals between the mechanistic model prediction and actual pressure loss due to the step input (Conlin *et al.*, 1997).

When the performance of the series structure was considered, it was seen that the structure performs significantly better. This was likely due to the neural network predicting new parameters at each time step which would be able to account for the step input (Conlin *et al.*, 1997). Thus, if one expects a system to undergo significant changes as time progresses it would be feasible to consider a series hybrid structure over a parallel structure. This is due to the parallel structure possibly not being able to extrapolate data well which lies outside the scope of the training data (Conlin *et al.*, 1997). It can also be inferred that if the mechanistic model describes the process well, then a series configuration provides more accurate predications than a parallel configuration (Conlin *et al.*, 1997).

In the work performed by Lee *et al.* (2002) hybrid modelling techniques were used to model an industrial coke-plant wastewater treatment process. The performance of a series and a parallel hybrid model in modelling the process were compared to the performance of the mechanistic model. An ANN was used as the data-based model in both hybrid model structures. An ANN was also trained to obtain the process output with the performance compared to the previously mentioned model configurations.

The mechanistic model was used to estimate four key parameters of the process. The mechanistic model was used to predict process data and was compared to actual measured data which corresponds to the respective input data. It was seen that the mechanistic model described some of the data well, however there were deviations between the actual data and the predictions. The deviations were particularly significant during a time duration where the process underwent a significant change (Lee *et al.*, 2002).

When the performance of the ANN was considered, it was found that the model does not predict the process data during the period of significant change well. This could be due to the training data not including the period of significant process change (Lee *et al.*, 2002). When the series hybrid model performance was considered, it was seen that the hybrid model did not improve on the predictive capabilities of the mechanistic model (Lee *et al.*, 2002). This reinforces the notion that a series hybrid model does not significantly improve the performance if the mechanistic model is mismatched to the system.

The parallel hybrid model did show an increase in performance due to the ANN sufficiently describing the residuals of the process (Lee *et al.*, 2002). This showed that the residuals of the process accounts for the dynamics of the system. By the residuals accounting for the system

dynamics the parallel structure partially accounts for the mechanistic model mismatch (Lee *et al.*, 2002). Thus, if a proposed mechanistic model does not describe a process well a parallel hybrid model would be preferable as it could increase the predictive performance of the first-principle model.

3.6 Hybrid modelling using recursive PLS

In the work performed by Jia et al., a copper extraction process was modelled using hybrid modelling incorporating recursive PLS (Jia, Mao, Chang & Zhao, 2011). The model structure proposed in the paper was an adaptive hybrid model which contains a recursive PLS model in parallel to the mechanistic model with an additional rectification model in parallel to the parallel hybrid model. The performance of the adaptive model was then compared to the performance of the mechanistic model and to hybrid models which contain a standard PLS model and a recursive PLS model respectively.

The adaptive model structure is shown in Figure 11 below. The model rectification section is responsible for compensating for the error between the model predictions and lab analysis values by adding the overall offset to the model prediction (Jia *et al.*, 2011). The offset of the model ($e(t)$) is determined by the weighted sum of the current bias of the model ($e_0(t)$) and the overall offset of the model at the previous time step ($e(t-1)$) as shown in Equation 43 where ω represents the weight parameter (Jia *et al.*, 2011).

$$e(t) = \omega e_0(t) + (1-\omega)e(t-1) \qquad [\ 43\ ]$$

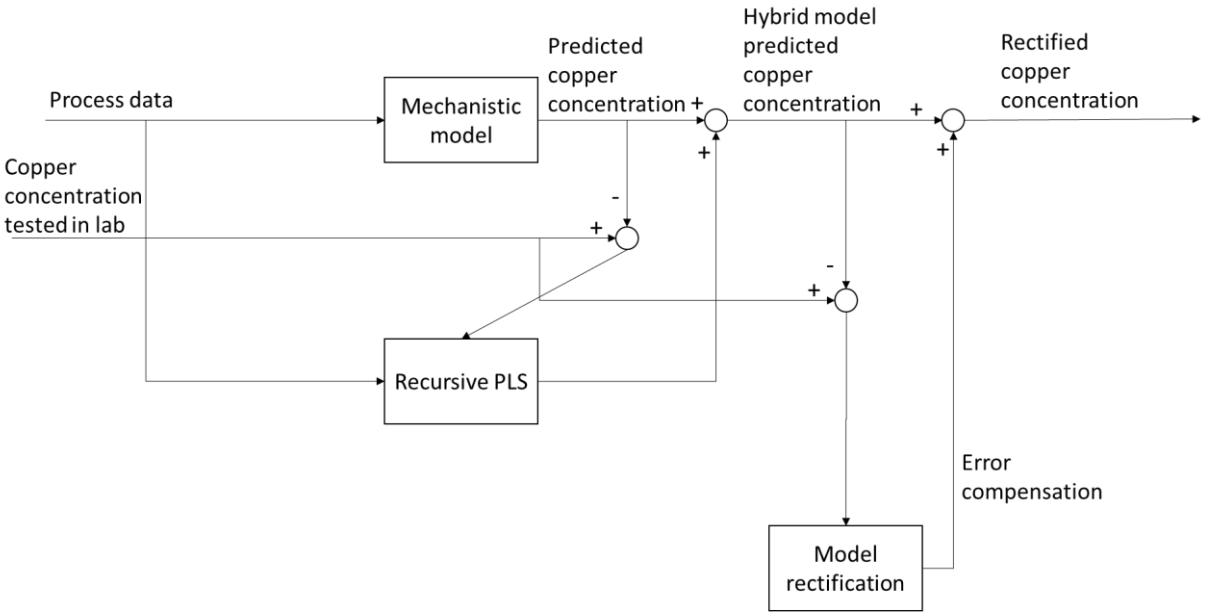

***Figure 11: Adaptive hybrid model structure as adapted from Jia et al.*(Jia *et al.*, 2011)**

The model rectification was introduced due to the extraction process having many factors which could not be accounted for as well as the uncertainty in the output values attained by lab analysis (Jia *et al.*, 2011). Thus, if a standard hybrid model is satisfactory in describing the behaviour of the process the adaptive component is not necessary as introducing an additional model component increases the execution time of the model. An additional issue with implementing the model correction component is that by incorporating sample analysis the model correction is in fact time lagged and not a true estimate or representation of the real time behaviour of the system (Jia *et al.*, 2011).

When the performance of the adaptive hybrid model was compared to the mechanistic model, PLS hybrid model and recursive PLS model it was found that the adaptive model had the best performance (Jia *et al.*, 2011). This was expected as the rectification section is specifically designed to reduce model error and offset. This along with the previously discussed work shows that there are many ways of improving the performance of standard hybrid models. The improvements are implemented on a case-to-case basis based on the model structure and information available. The adaptive model provides a feasible way of improving the model performance if the standard hybrid model structures are not satisfactory in their performances however the performance will need to be tested and scrutinised (Jia *et al.*, 2011). The model

rectification is also not specific to the PLS empirical model and thus can be applied to hybrid models containing any empirical model.

As the rectification model component receives input from the sample analysis section based on the analysis results each day the rectification section constantly improves the performance of the model. An advantage of the adaptive model is that the rectification calculation implemented is a simple calculation and thus implementing this in the model should not greatly impact the execution time of the model (Jia *et al.*, 2011). The time lagged nature of the model could also possibly present a challenge in the implementation of the model with regards to a system with constantly changing or rapidly changing process conditions (Jia *et al.*, 2011).

3.7 Evaluation of Literature

In the field of fault detection in process monitoring many different models and techniques have been developed to suit a wide array of process characteristics. Knowledge-based fault detection techniques require expert knowledge and experience of a particular process and as such is only applicable to a narrow range of processes (Miljković, 2011). As such the development of a knowledge-based fault detection technique will not be pursued. The two remaining sub-divisions of fault detection based on the categories found in Miljković (Miljković, 2011) are model-based fault detection methods and data-driven methods. Fault detection methods which incorporate machine learning and statistical techniques would be classified as data driven methods.

Data-driven methods which use machine learning techniques at times pose fault detection as classification problems. In classification problems the process data is categorised as either belonging to a normal operation class or a faulty operation class. As the classification problem only comprises of two classes binary classifiers are used.

Fault detection methods which utilise kNN as the fault detection method have shown high fault detection accuracy which indicates that kNN based fault detection is a viable method (He & Wang, 2007). A significant advantage of kNN models is the simplicity of understanding and implementing the model. Another advantage is that the model is easily expanded to multiclass classification (He & Wang, 2007). A significant drawback of the model is the computing cost involved with implementing the model. This is due to the model having to compare each new observation to the entire training dataset which can be time consuming to execute, especially with larger training sets (He & Wang, 2007).

Fault detection methods utilising ANNs have been evaluated and it was found that the technique showed both a high detection accuracy and low false alarm rate (Heo & Lee, 2018). The ANN technique shows good performance and can be applied to a large range of datasets both linear and non-linear in nature. ANN classifiers can also easily be extended to multiclass classification (Heo & Lee, 2018). An important note is that introducing neural networks models can rapidly increase the complexity of a proposed method as neural networks are not always easily interpretable. The computational cost of training a neural network can also become great depending on the architecture of the ANN.

Fault detection models utilising SVMs have seen use and have shown good performance when applied to the Tennessee Eastman process (Yin *et al.*, 2014). The use of kernel functions in the SVM model allow the classifier to perform well on non-linear datasets. SVM classifiers do not have as large a computational cost involved with classifying new observations. This due to the data entries only needing to be compared to the location of the support vectors and not the entire training dataset. A possible drawback of the method is that SVM classifiers are designed to be binary classifiers. Multiclass classification is possible using SVM classifiers however this requires adjustments to the classification method and the training of additional SVM classifiers (James *et al.*, 2017).

One method implemented to attempt to improve the performance of the fault detection methods which use machine learning techniques is to make use of feature extraction or selection in the data pre-processing stage. The extracted (or selected) features are fed to the classifier and used for the fault detection as was done by Onel et al. (Onel *et al.*, 2019) where a feature selection method was coupled with a SVM fault detection method. This leads to the idea of obtaining additional information from the available process data and using it to increase the performance of a fault detection and identification technique.

A possible method of obtaining the additional information could possibly be derived from the principles behind model-based parameter estimation fault detection. In model-based parameter estimation a model is used to describe a process and process data is used to estimate model parameters which are then used for fault detection (Miljković, 2011). This was done in the work by Che Mid and Dua (Che Mid & Dua, 2017) where the parameter estimation is performed by using multiparameter programming which uses incoming process data. The fault detection using parameter estimation showed a high fault detection rate (Che Mid & Dua, 2017). A drawback

of the model-based method is that the considered process model should describe the process data and if there is significant model mismatch the fault detection will not perform well at all (Che Mid & Dua, 2017). Also performing a direct regression on incoming data online to obtain parameter estimates would be computationally intensive.

Hybrid modelling could possibly be used as the parameter estimation method as the very nature of hybrid modelling includes the use of first-principle process models. As such hybrid modelling could be viable as a means of obtaining model parameters from incoming process data. An adjusted series structure is a possible option as series hybrid modelling involves using data-based models to estimate model parameters from incoming process data (von Stosch *et al.*, 2014). These estimated parameters are then fed to a first-principle models. A parallel structure would not be feasible as in the parallel structure the data-based model is used to estimate the model output residuals. These residuals are then used to increase the accuracy of the model predictions (Bhutani *et al.*, 2006). A modified series structure could be applicable, where the data-based model is trained using estimated model parameters and process data. The data-based model is trained so that the model parameter estimates can be obtained from incoming process data. There are several data-based models which could implemented in the hybrid model.

Some of the possible data-based models seen in the literature are ANN, SVR, PLS and recursive PLS. ANN has been shown as a viable data-based model in hybrid modelling in the work of Bhutani et al. (Bhutani *et al.*, 2006). The SVR data based model has successfully been applied to hybrid modelling as seen in the work of Hu et al. (Hu *et al.*, 2011). The SVR model does show good performance in parameter estimation however it could possibly increase the complexity of the parameter estimation model. The PLS model is a possible data-driven model which could be used as the parameter estimator. The PLS model operates as a multivariate regression and dimensionality reduction method (Geladi & Kowalski, 1986). The PLS method also does not have a high computational cost involved with online parameter estimation. The use of dynamic PLS is a possibility as by considering a range of process observations which encompass an entire dynamic cycle could lead to more accurate parameter estimates (Ricker, 1988). The recursive modification of the standard PLS model is also considered. By incorporating parameter residuals recursively and updating the PLS model one could possibly obtain more accurate parameter estimates which could lead to improved classification performance (Jia *et al.*, 2011).

As seen hybrid modelling provides a way of developing models which have improved accuracy and performance, and can also be used to retroactively improve developed models. This is achieved using system performance data which allows the developed model to mimic more closely the observed and recorded real world performance of the system. Hybrid modelling could be used to estimate additional information based on incoming process data (Conlin *et al.*, 1997). This information could be used to improve the performance of a fault identification and classification model.

As fault identification involves identifying the appropriate fault which caused an alarm from different possible fault scenarios, fault identification can be posed as a multiclass classification. A simultaneous fault detection and identification method can be developed using the multiclass approach if normal operating conditions are specified as one of the classes as was done by Heo and Lee (2018). The multiclass adaption of the SVM classifier is considered to address this problem. This was done due to the ability of the SVM classifier to handle non-linearly separable datasets. In order for the possible fault identification method to handle non-linearly separable datasets the RBF kernel is selected as the kernel function. This due to the generalization ability of the RBF kernel (Bishop, 2006). The performance of the classifier could possibly be increased by feeding additional information in the form of model parameters estimated from the process data to the classifier. This could possibly be viewed as a form of feature extraction.

These key takeaways lead to the development of a potential simultaneous fault detector and identifier. The proposed method will utilise hybrid-modelling based parameter estimation, with parameter estimation based on the model-based parameter estimation as seen in parameter estimation fault detection. The classifier proposed for the fault identification is a multiclass SVM. The performance of the classifier will be evaluated when both the RBF kernel is applied as well when no kernel transformation is used. The classifier will be fed parameters estimated by the hybrid modelling technique will be fed to classifier along with the process data and the performance will be evaluated. The performance will conclude whether the additional parameter data does improve the fault classification. The PLS model along with the dynamic and recursive variations of the model are considered as the data driven models in the hybrid model and the performances of the models will be compared. The fault identifier will be tested on a test case continuous stirred tank reactor (CSTR) model similar in nature to the CSTR model implemented by (Addo, 2019).

4 METHODOLOGY

The fault identification problem posed in this work can be treated as a multiclass classification problem. Based on this a multiclass classifier is chosen as the primary fault identification model. An SVM classifier adapted for multiclass classification is selected as the fault identifier. This is due to the SVM classifier being a robust classifier which performs well on many different datasets. The kernel SVM classifier is considered due to its ability in classifying non-linearly separable data. This a key feature as many process faults form part of non-linearly separable datasets.

The performances of both standard and kernel SVM fault identification models are compared. In order to improve the performance of the fault identifier a hybrid modelling feature extraction method is suggested. A hybrid model will be used to extract model parameters from the available dataset. The parameters are then fed to the classifier alongside the dataset. By having additional information available the SVM classifier could possibly show an increase in performance as the model parameters could possibly reflect the process fault more clearly. This would result in clear class boundaries being present thus increasing the fault classification performance.

Two hybrid model structures are considered for the feature extraction method, namely a shortcutting hybrid model structure and a recursive hybrid model structure. Both proposed hybrid model structures are altered series hybrid model structures. The shortcutting hybrid model makes use of standard PLS regression and dynamic PLS regression models while the recursive hybrid structure makes use of the recursive PLS model. The models are tested on a non-isothermal continuously stirred tank reactor (CSTR) model upon which three different faults are implemented. The performance of the hybrid models is compared to the performance of a fault identification models which only uses SVM and kernel SVM classifiers. The chapter describes the process of developing the test case CSTR model and the faults applied to the model. The chapter also describes the hybrid model structures and how they are applied for fault identification.

4.1 CSTR Model

The CSTR model developed as the test case model is a jacketed non-isothermal CSTR reactor with one reaction with no side reactions producing by-products. The CSTR reactor is illustrated in Figure 12 below. The reaction is a first order exothermic, irreversible reaction as shown by the chemical equation in Equation 44 below:

$$A \rightarrow B \qquad [\ 44\ ]$$

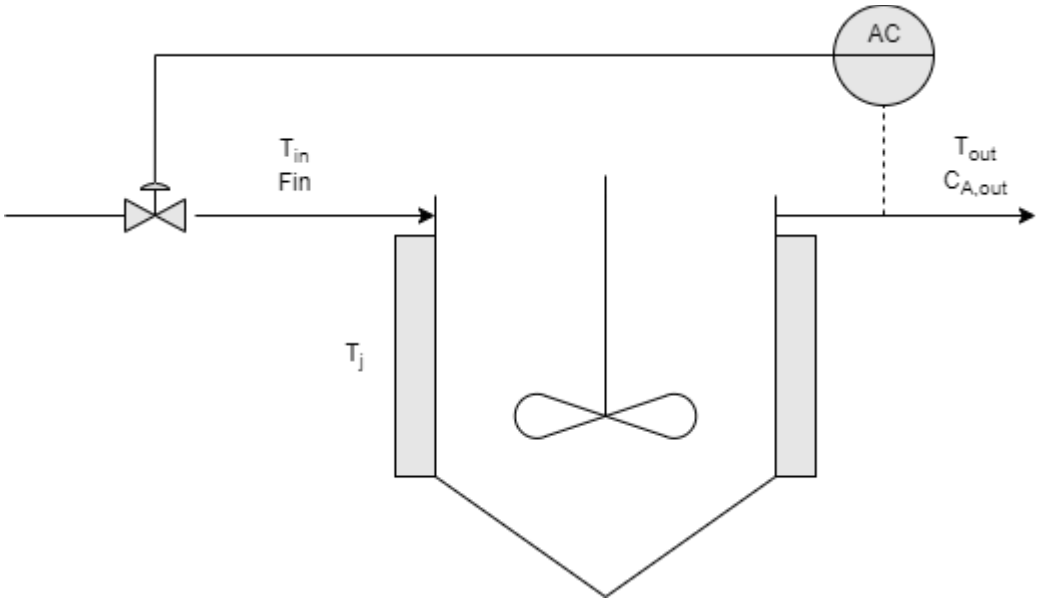

Figure 12: Jacketed CSTR with a feedback concentration controller

The assumptions made during the development of the model are as follows:

- The reactor is well mixed
- The system volume remains constant due to the reactor having an overflow stream
- There are no dead zones in the reactor
- The fluid densities and heat capacities remain constant
- The constant pressure and constant volume heat capacities are identical
- The reactor jacket is an infinite reservoir

Based these assumptions and reaction the CSTR model is represented by the following pair of ordinary differential equations (ODEs) as shown in Equations *45* and 46 which describe the component balance of A and the energy balance of the reactor respectively (Seborg *et al.*, 2010). The model equations describe the change in outlet concentration ($C_{A,out}$) and the temperature of the outlet stream (T_{out}) with respect to time.

$$\frac{dC_{A,out}}{dt} = \frac{F_{in}}{V}\left(C_{A,in} - C_{A,out}\right) - k_0\left(e^{\left(\frac{Ea}{RT_{out}}\right)} \times C_{A,out}\right) \qquad [\ 45\ ]$$

Where $C_{A,out}$ represents the outlet concentration of A, t represents time, F represents the inlet volumetric flowrate, V the reactor volume, $C_{A,in}$ the inlet concentration of A, k_0 the pre-exponential factor, E_a the activation energy, R the universal gas constant and T_{out} the outlet/reactor temperature.

$$\frac{dT_{out}}{dt} = \frac{F_{in}}{V}(T_{in} - T_{out}) - \frac{\Delta H_r}{\rho \times c_p} \times k_0\left(e^{\left(\frac{Ea}{RT_{out}}\right)} \times C_{A,out}\right) - \frac{UA}{\rho \times c_p \times V} \times (T_{out} - T_j) \qquad [\ 46\ ]$$

Where T_{in} represents the inlet temperature, ΔH_r represents the heat of reaction, ρ the fluid density, c_p the constant pressure heat capacity, UA the overall heat transfer coefficient and T_j the jacket temperature. The model inputs are the inlet flow rate (F), the inlet temperature (T_{in}), and the jacket temperature (T_j) with the initial values of each given in Table 1 below.

Table 1: Initial values of model inputs

VARIABLE	VALUE	UNIT
F	100	L/min
T_{IN}	350	K
T_J	290	K

The CSTR model is implemented using the parameter values listed in Table 2. The parameters are adapted from a test case model found in Seborg et al. (Seborg *et al.*, 2010).

Table 2: Model parameter values

PARAMETER	VALUE	UNIT
K_0	7.2×10^{10}	1/min
$C_{A,in}$	1	mol/L
E_A	-72 747.5	J/mol
R	8.314	J/(mol·K)
V	100	L
ΔH_r	-50 000	J/mol
UA	50 000	J/(min·K)
$\rho \times c_p$	237	(mol·J)/(m^3·K)

The jacketed non-isothermal CSTR is selected as the test case model due to it being a widely used model which is applied to many different process systems and is easily understood. The CSTR model, despite appearing simple, can be easily adjusted to increase the model complexity. The non-isothermal CSTR model is non-linear with respect to certain model output values such as the outlet stream temperature. The non-linearity of the model allows the performance of the hybrid model fault identification techniques on non-linear systems to be evaluated. The model contains several process parameters which are set and remain constant during the simulation. Due to the nature of the model several different process faults can be applied to the model. The process faults are implemented such that they impact and alter the process parameters. The fault laden as well as normal operation scenarios are used to generate datasets upon which the fault identification models are then applied.

4.2 Model implementation and system faults

The model is built up and implemented in Simulink with the applied faults being adapted from the fault laden CSTR model developed by (Addo, 2019). The Simulink model includes feedback control implemented on the inlet flow rate to maintain the outlet concentration at a

predetermined setpoint. Under normal operating conditions and unless otherwise specified all process inputs remain at initial conditions. An in-depth description of the Simulink model is given in Appendix A.

In order to generate a dataset which more closely mimics a dataset obtained from a real-world process, only select process variables were recorded. The variables were selected to represent important plant measurements which would be recorded in a real-world process to further emulate real word conditions. The measured variables were sampled and recorded every minute. In real-world processes, concentration readings are determined by analysing samples. The samples are taken at certain times during the day or upon request with the concentration of the sample only available once the analysis is complete resulting in a delay concentration values. It is assumed that the outlet concentration is measured instantaneously in the same way that the flow rate or temperatures are recorded. It is noted that this would not be possible in an actual process but is assumed for the purpose of this study. This was done to have real time concentration values which can be used in the fault identification. The selected measured variables are shown in Table 3 below. To further emulate plant conditions variable noise is added to each measurement to replicate noisy plant measurements. It is assumed that the noise would be normally distributed and the noise is implemented as an autoregressive function (Addo, 2019).

Table 3: Recorded process measurements

VARIABLE	UNIT
F	L/min
$C_{A,OUT}$	mol/L
T_I	K
T_J	K
T_{OUT}	K

A feedback proportional integral (PI) controller is implemented in the CSTR model in order to maintain the outlet concentration of A near a predetermined setpoint. A block diagram

representing the feedback control is shown in Figure 13. The control is implemented by monitoring the outlet concentration of A ($C_{A,out}$) and adjusting the inlet flow rate F as required. Applying a simple control scheme to the process simulation results in a simulation model which more closely resembles a real-world process as a real process would implement control to maintain product quality. This in turn generates data which more closely mimics process data which one would be able to obtain from a real-world process.

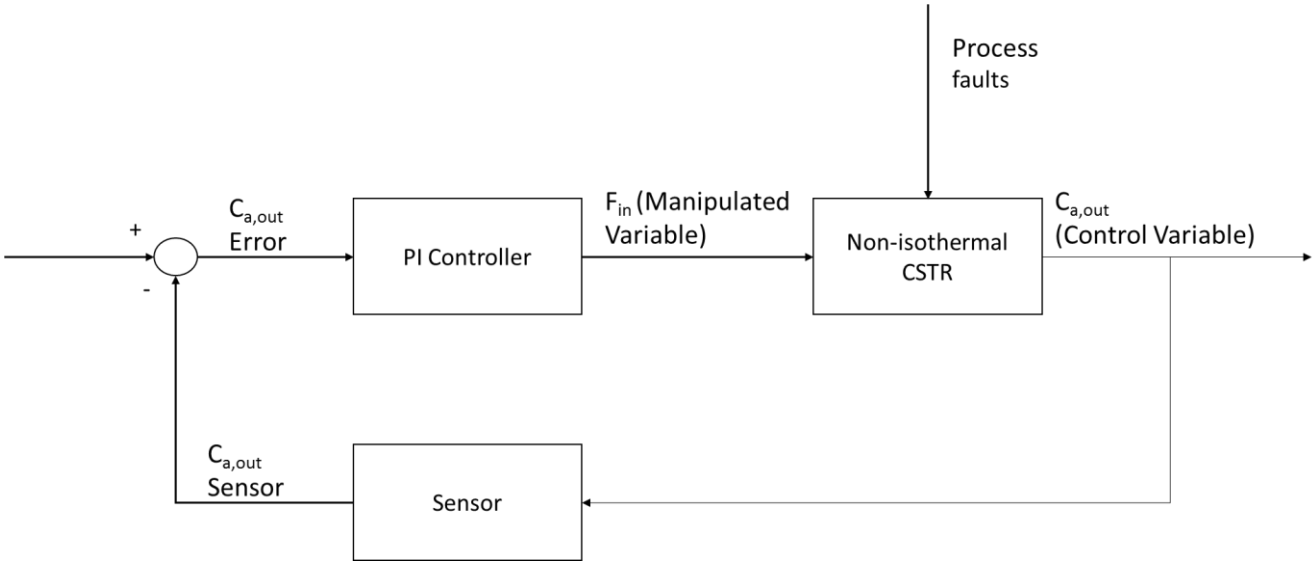

Figure 13: Non-isothermal CSTR PI feedback control block diagram

The outlet concentration of A is selected as the measured variable as one would want to maintain outlet product quality at some value. The set point for the outlet concentration is set as 0.955 mol/L. This value was obtained by considering the outlet concentration of the simulation at normal operation without the controller applied. The outlet concentration values are then averaged over a selected time window. The inlet flowrate is selected as the control variable as the inlet flowrate affects both the amount of A entering the reactor as well as the temperature of the reactor. The temperature of the reactor in turn affects the reaction rate and the outlet concentration of A. The inlet flowrate also represents a process variable which could be adjusted by some control action and would also be a process variable which would be selected as a control variable in a real-world process. In a real world process the control action would be implemented on some valve which would result in a change in the inlet flowrate. The valve opening percentage or position could be viewed as the control variable however for the purpose of this model the flowrate is selected as the control variable and is adjusted directly. The control parameters for the controller are given in Table 4. The controller parameters given

are not the optimal controller parameters and were set as they provided an adequate controller response action.

Table 4: PI controller parameters

VARIABLE	VALUE
P	5
I	50

Three different process faults are applied to the CSTR model namely, a catalyst deactivation fault, an inlet concentration decline fault, and a heat transfer fault. Mathematically the process faults are implemented as ramp decline faults as shown in Equation 47.

$$P_t = P_{t-1} - mag \times P_{NOC} \times \Delta t \qquad [\ 47\ ]$$

Where P_t is the process parameter at the current time step, P_{t-1} represents the process fault at the previous time step, P_{NOC} represents the process parameter under NOC, Δt represents time step magnitude and mag represents the fault ramp magnitude. The process fault magnitude is expressed as a percentage of the parameter value at NOC and describes the rate at which the process parameter decreases from the NOC value per minute.

The catalyst deactivation fault is implemented to emulate the gradual reaction rate decrease due to a decrease in catalyst activity which occurs over time. As such the catalyst deactivation fault affects the reaction rate and does so by effecting the reaction pre-exponential factor (k_0). In a real process the decline in catalyst activity could be due to several factors such as catalyst fouling or catalyst denaturing (Fogler, 2006). The fault is implemented and evaluated at different fault magnitudes.

The effect of the fault on the reaction rate ($k_0 \times e^{\left(\frac{E_a}{RT_{out}}\right)}$) is illustrated in Figure 14. The reaction rate plot is generated when a fault magnitude of 62.5% is applied to CSTR model. As seen, there is significant decrease in the reaction rate once the error is applied to system

resulting in reaction rate of 0 min^{-1}. This is a not a realistic process fault as such a decrease in reaction rate would not be seen in a real process and is implemented just to evaluate the performance of the fault identification method. It is assumed then when the appropriate corrective measure is applied to the remedy the fault the process immediately returns to normal operating conditions.

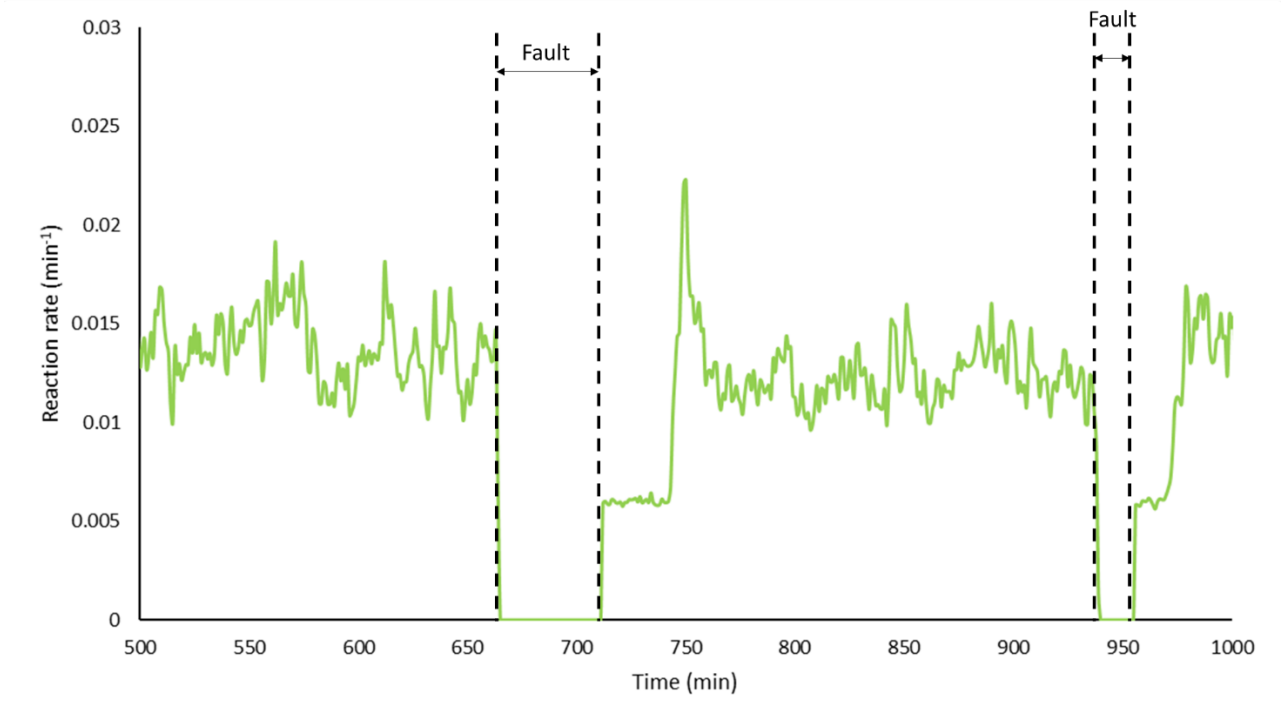

Figure 14: Reaction rate for a time window from 500 - 1000 minutes

The inlet concentration decline fault emulates changes in the quality of inlet stream. The quality decline effects the inlet concentration of reagent A ($C_{a\,in}$). The inlet concentration decline fault mimics a decrease in the inlet concentration of A which could be due to some upstream disturbance. The inlet concentration of A is a process variable which is neither measured in the simulation nor will be obtained by the parameter estimation. Thus, by implementing this fault it will evaluate the performance of the fault identification method in identifying a fault applied on a non-measured or estimated parameter. The effect of the inlet concentration fault is shown in Figure 15 below.

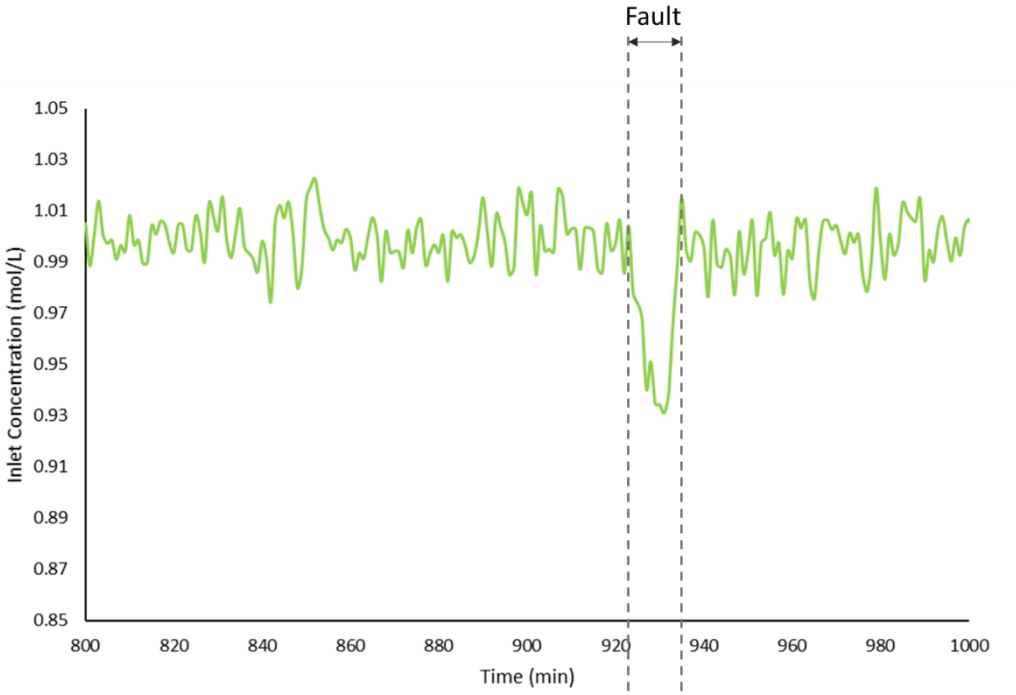

Figure 15: Effect of inlet concentration fault over a time from 800 - 1000 minutes

The heat transfer coefficient fault mimics a decrease in the rate heat is transferred from the CSTR tank to the cooling jacket surrounding the tank by transferring heat across the wall of the reactor. The decrease in heat transfer rate in turn effects the heat transfer coefficient. In a real process this decrease in heat transfer could be viewed as being caused by fouling on the reactor wall. The fouling would in turn increase the resistance to heat transfer thus decreasing the overall heat transfer coefficient. The effects of the fault are shown in Figure 16.

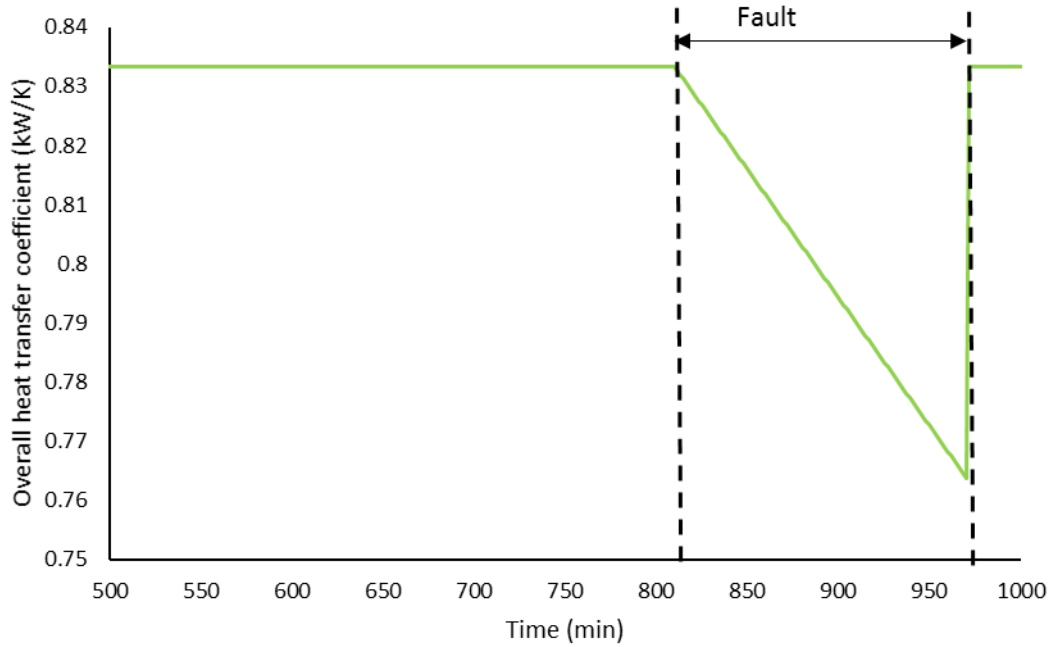

Figure 16: Effect of overall heat transfer coefficient fault for a time window from 500 - 1000 min

These faults are chosen as the effects of the faults have on the system are not extremely large and have effects which cascade through the process. The faults are randomly implemented with mean durations and duration variance for both normal operation and fault conditions specified. It is assumed that both normal operation and fault operation durations are normally distributed. The faults are applied at different magnitudes to observe the performance of the fault identification models as the faults become larger and more prominent. The process faults along with the magnitudes, durations, and variance for each are given in Table 5 and the duration for normal operation between each fault instance is given in Table 6. The fault magnitudes given in Table 5 are the fault magnitudes (mag) used in implementing the process faults using Equation 47. The smallest fault magnitude represents the smallest fault magnitude considered; the largest fault magnitude is the largest considered with the fault magnitude step representing the magnitude of the difference between each fault considered. For example, if the heat transfer fault is determined based on the values given in Table 5 the range of fault magnitudes (and thus the values of mag) are 0.15,0.152,0.154,0.156,0.158 and 0.16.

Table 5: Process fault magnitudes, durations, and variance

Process fault	Smallest fault magnitude (%)	Fault magnitude step (%)	Largest fault magnitude (%)	Mean duration	Duration variance
Catalyst deactivation	6.94	1.39	62.5	80	25
Overall heat transfer coefficient fault	0.15	0.002	0.16	200	25
Inlet concentration fault	0.11	0.021	0.84	12	3

Table 6: Normal operation durations for each process fault

Variable	Duration	Duration variance
Catalyst deactivation	600	150
Overall heat transfer coefficient fault	600	150
Inlet concentration fault	800	130

4.3 Parameter estimation

When the fault detection models are applied to the training datasets obtained using the CSTR model, some model parameters are regressed using the training data. The regressed parameters represents features extracted from the training dataset. Although all the parameter values are set when generating the datasets, it is assumed that some parameter values are unknown when applying the fault identification models. Thus, modified model equations are used as shown in Equations 48 and 49 below where certain parameters are grouped together and estimated. The grouped parameter terms comprise of the unknown parameters (k_0 and UA) grouped with associated constants to make up one term in the equation. The unknown model parameters which are estimated are the reaction rate (k) and the grouped heat transfer coefficient (UA). The reaction rate comprises of the grouped $k_0 \times e^{\left(\frac{E_a}{RT_{out}}\right)}$ term and the grouped heat transfer coefficient represent the $\frac{UA}{\rho \times c_p \times V}$ term.

$$\frac{dC_{a,out}}{dt} = \frac{F_{in}}{V}\left(C_{a,in} - C_{a,out}\right) - k \times C_{a,out} \qquad [\ 48\ ]$$

$$\frac{dT_{out}}{dt} = \frac{F_{in}}{V}(T_{in} - T_{out}) - \frac{\Delta H_r}{\rho \times c_p} \times \left(k \times C_{a,out}\right) - UA \times (T_{out} - T_j) \qquad [\ 49\ ]$$

The estimated parameters then become additional features included in the process dataset. The catalyst deactivation fault and heat transfer fault thus have features on which the faults are directly applied. The inlet concentration fault has no associated feature on which the fault is directly applied.

The parameters are estimated using non-linear regression in MATLAB where the regression is performed using the lsqnonlin function in MATLAB. The parameter estimates are then used to train the data-based model components of the hybrid models to predict the parameter values based on incoming data. This is done as non-linear regression can be a time-consuming process especially with a large set of inputs if real time online parameter estimation is desired or required.

The non-linear regression is implemented over a sliding window by considering each observation within the window. The regression using a sliding window as is illustrated in Figure 17. A sliding window is a window of predetermined length which is applied to a dataset. The window indicates a subset of the data on which actions and mathematical operations are performed. The window moves sequentially through the data once the desired actions are performed on the entries in the current window. The new starting point of the window sequentially follows the previous start point (i.e. start moving from t_1 to t_2). In this work a window length of 16 minutes is used as this twice the process time constant thus should be able to effectively capture the dynamics.

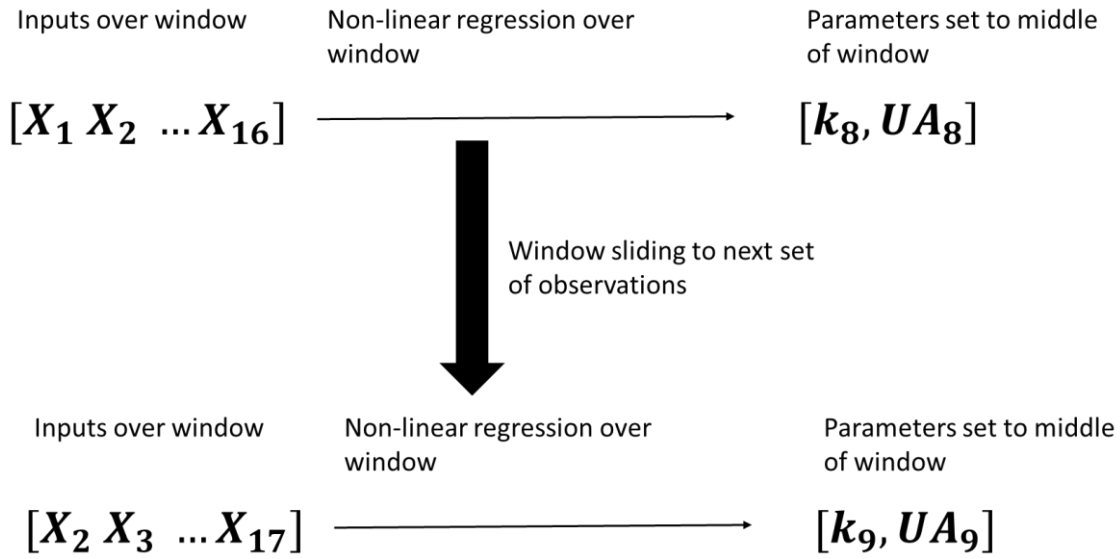

Figure 17: Sliding window implementation of non-linear regression

The parameter regression estimates are obtained by solving the model ordinary differential equations. The differential equations are solved over the time window using a differential equation solver. There are several different equation solvers one can use depending on the equations and data available. The ode45 ordinary differential equation solver is selected due to its versatility. Initial parameter estimates are set for the unknown model parameters based on available knowledge of the system. The parameter estimates are then used to solve the differential equations by setting the estimates as the respective parameters. This done for each time instance within the window.

The outputs from the differential equations (T_{out} and $C_{a,out}$) are compared to the actual CSTR outputs at each time instance within the window. The squared error between the equation outputs and the CSTR outputs at each time instance is determined. The mean of the squared

errors is then taken over the entire time window. The mean squared error for the time window (i.e. the objective function) is minimized by adjusting the parameter estimates.

The parameter estimates obtained through the regression are then set as the estimates for the time instance in the middle of the window. The new initial parameter estimates are set as the parameter estimates obtained by the previous error minimization. Once the window moves to the next input start point the regression is performed again. The entire regression process is repeatedly performed until the end point of the time window reaches the last available data point.

4.4 Hybrid model shortcutting structure

The first model structure considered for the fault identification models is the hybrid model shortcutting structure. The hybrid model shortcutting model is based on an alternate series arrangement in which the first-principle model is first followed by a black-box model. The hybrid model shortcutting structure is shown in Figure 18 and comprises of two components namely an offline and an online component. The offline component contains the first-principle model, and the online component contains both the data-based model and the SVM classifier.

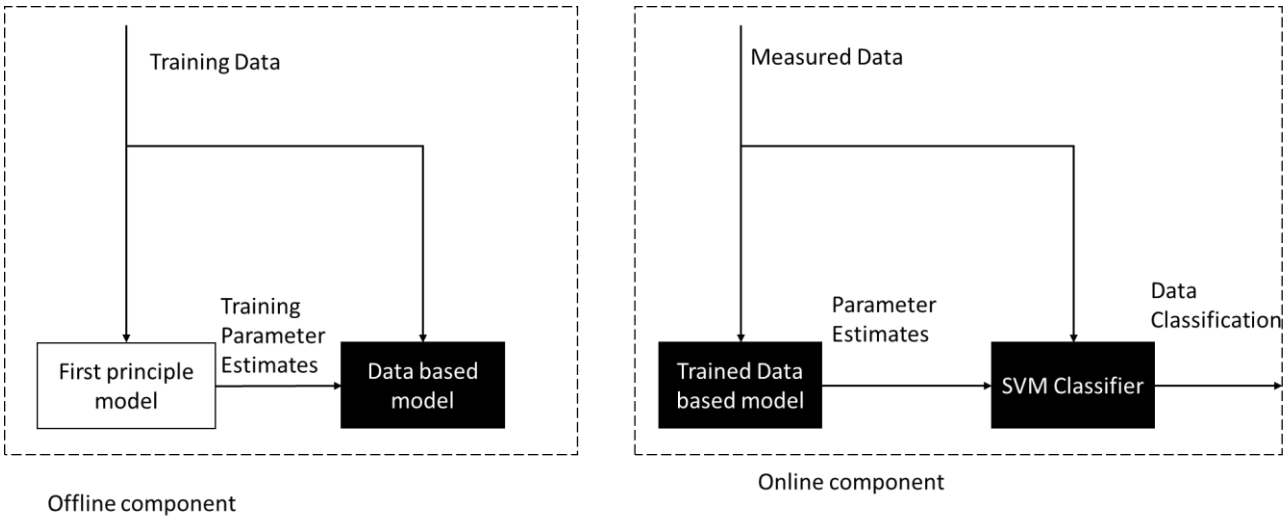

Figure 18: Hybrid-modelling shortcutting combination model structure

The standard hybrid model involves estimating parameters from the measured data using a first-principle model by using non-linear least squares regression. Estimating these parameters continuously by using the first-principle model can become time consuming due to the continuous regression taking place. It is proposed that an initial set of parameters be estimated

by the first-principle model with the parameters then used to train a data-based model. The data-based model is then used to predict the parameter values based on incoming data. The first-principle model is only used in estimating the training parameters, if the shortcutting aspect is not implemented then the white-box model would be continuously used to estimate the process parameters by non-linear regression. The SVM classifier is then trained using process data as well as using parameter estimates estimated by the trained data-based-model. The regression of the parameters using the first-principle model and the training of the data-based model and SVM classifier comprises the offline component of the hybrid model.

The data-based model is used to estimate the parameters continuously using incoming measured data. This is the shortcutting aspect of the model as it removes the need to obtain the parameter estimates via non-linear regression. The data-based model parameter estimates are then fed to the SVM classifier for fault identification. The online component of the model consists of estimating the model parameters using the data-based model and identifying the fault using the SVM classifier. The performance of this model structure is evaluated using two different data-based models namely the ordinary PLS model and the dynamic PLS model.

4.5 Recursive hybrid model structure

The second hybrid model considered is the recursive hybrid model structure. The recursive hybrid model follows the standard series hybrid model structure where a data-based model is implemented first followed by a first-principle model with recursive feedback to the data-based model. The recursive hybrid structure is illustrated in Figure 19 below which represents the online component of the hybrid structure.

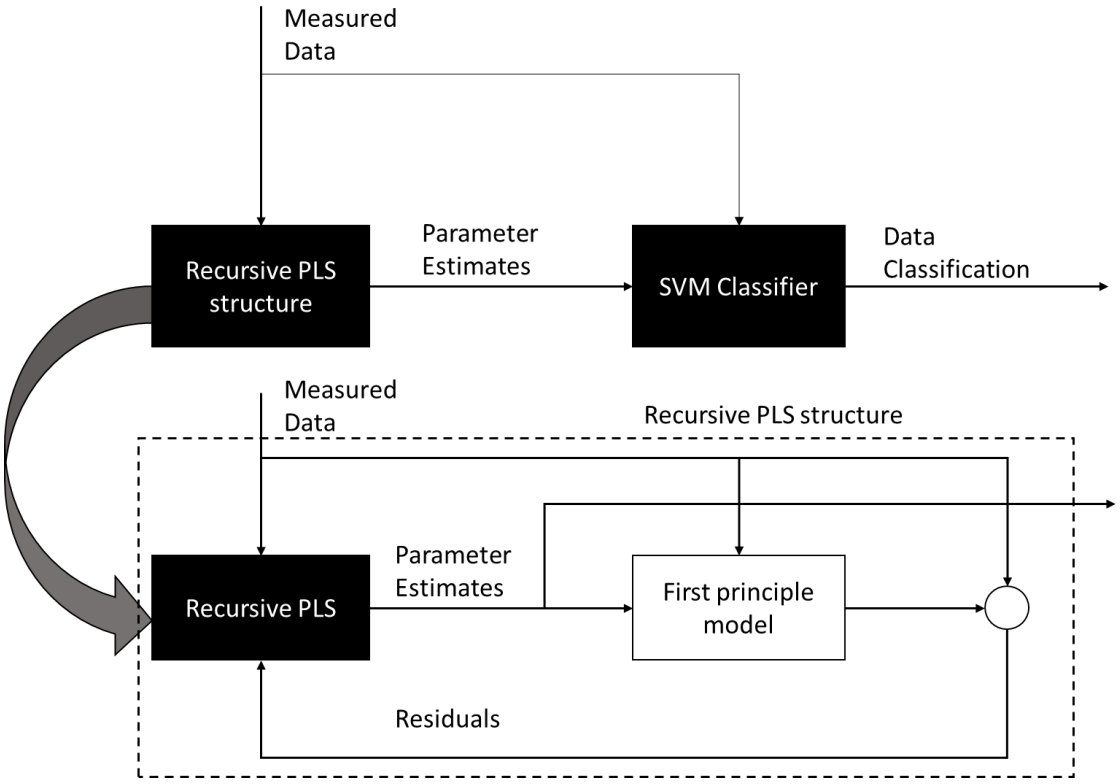

Figure 19: Recursive hybrid model structure

The data-based model trained using parameter estimates obtained from the training dataset predicts new parameter estimates based on incoming data fed to the model. These parameter estimates are then fed to the SVM classifier which, along with the measured data, is used to determine which fault condition the data corresponds to.

The recursive aspect of the model occurs when the parameter estimates are fed to the first-principle model which predicts the system outputs using the parameter estimates. The output predictions are then compared to the actual measured output data. The information obtained from the comparison (the residuals) are then recursively fed back the data-based model to update the model to increase the accuracy of the parameter predictions. The proposed data-based model for this hybrid structure is the recursive PLS model.

By continuously updating the data-based model the recursive model structure continuously adapts to process changes which occur over time. This eliminates the need to manually update the model parameters as the recursive model structure does this by design. This also ensures that the model remains up to date with the current state of the process which is advantageous if the model is applied as an online fault identification method. The frequency at which the

data-based model updates its parameters as well as the importance of historical data against new incoming data is controlled by adjusting the window size and weighting parameter of the model. A possible disadvantage of the recursive model is that if the system is running at fault conditions for an extended period the model would begin to recognise the fault state as normal operating conditions.

4.6 Support Vector Machine (SVM) classifier approach

As three different fault conditions are applied to the CSTR model the fault detection is considered a multiclass classification problem. The choice of an appropriate classifier is thus important to account for the multiple classes. As the model and outputs of the model are not linearly separable in nature a classifier which can perform well with non-linearly separable data is required. The one-vs-one multiclass SVM classification approach is followed. The SVM classifiers are trained pairwise on the process faults and normal operation conditions.

As SVM classifiers are kernel-based methods the choice of appropriate kernel is key to the effectiveness of the method as the classifier only works as well as the kernel allows it to. The kernel used for this method is the gaussian or radial basis function kernel. The gaussian kernel is chosen as it has wide applicability and performs well with many different non-linear datasets. The performance of SVM classifiers which implement the linear kernel (thus being a standard SVM model), and the RBF kernel are compared using the sensitivity and specificity performance measures.

4.7 Implementation approach

To aid in the discussion of the method implementation and the results of the study the PLS models, SVM classifiers and hybrid model combinations are assigned abbreviations as shown in Table 7.

Table 7: Models and model combination abbreviations

ABBREVIATION	MODEL
kSVM	Kernel SVM
r-SVM	SVM using non-linearly regressed parameters
r-kSVM	Kernel SVM using non-linearly regressed parameters
dPLS	Dynamic PLS
rPLS	Recursive PLS
PLS-SVM	SVM using PLS estimated parameters
PLS-kSVM	Kernel SVM using PLS estimated parameters
dPLS-SVM	SVM using dynamic PLS estimated parameters
dPLS-kSVM	Kernel SVM using dynamic PLS estimated parameters
rPLS-SVM	SVM using recursive PLS estimated parameters
rPLS-kSVM	Kernel SVM using recursive PLS estimated parameters

The model combinations and fault identifier implementations are illustrated in Figure 20. The diagram illustrates the flow of data between the generated datasets and the fault identification models as well as displaying the performance measures. The datasets considered in this work are generated by running the CSTR Simulink model for each operation mode for a duration of 4000 minutes. A dataset is generated for the normal operation mode as well as each fault mode where the fault is implemented at each of the fault magnitudes considered. In each fault mode only one of the process faults is applied with the frequency and duration of the fault being determined by the fault implemented. Each dataset is saved and then loaded for use as needed by each fault detection model. This ensures that all model performances are compared using the exact same dataset and fault conditions. This also reduces the execution time for each model by removing the need to generate each dataset by simulating the CSTR model for each run.

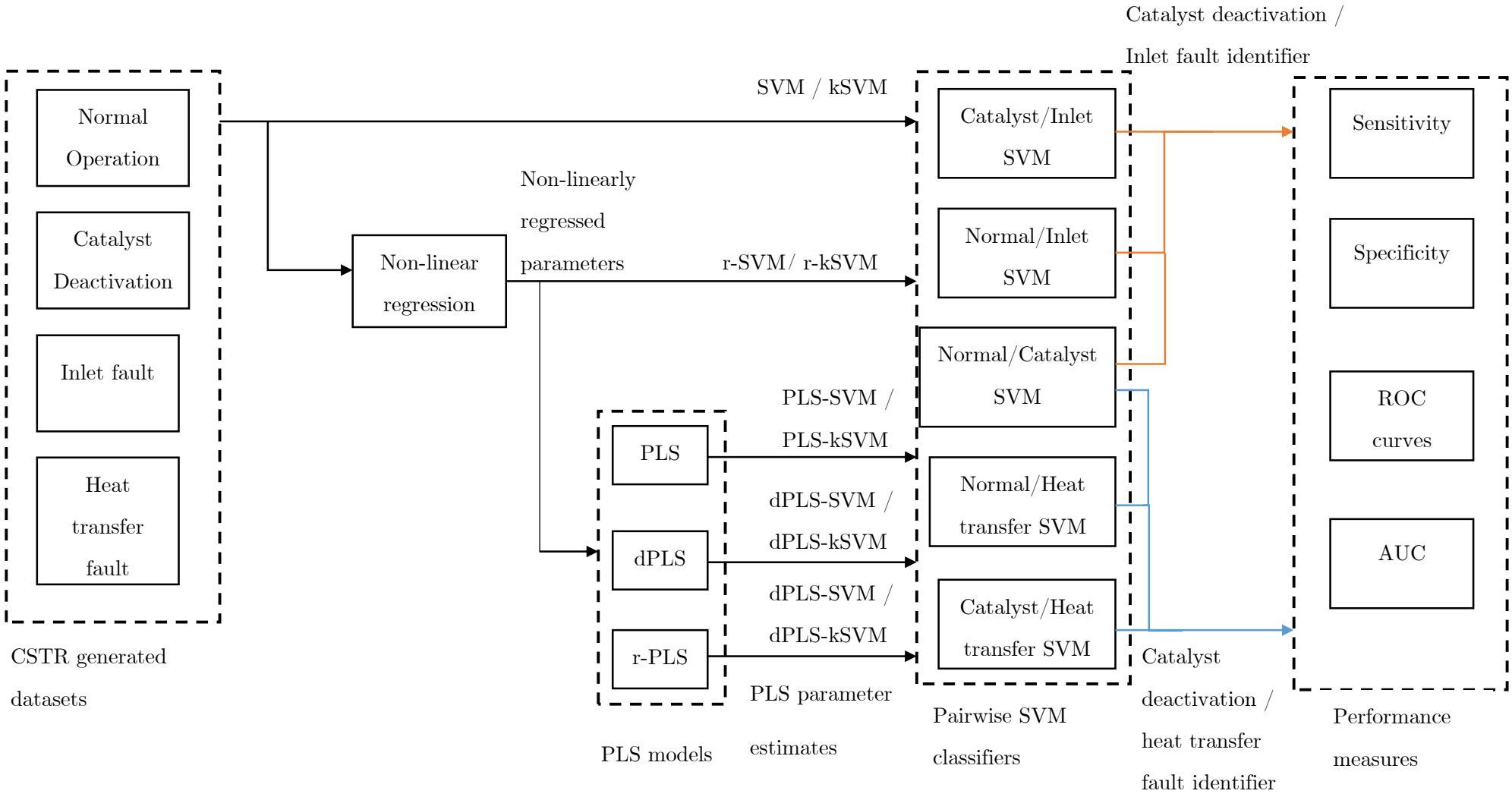

Figure 20: Study methodology illustrating flow data, model combinations and performance metrics

When the recursive PLS model is implemented an initial set of residuals is required as these account for the residuals for the first data block in the block-wise recursive PLS algorithm. These initial residual estimates are obtained by feeding the parameter estimates from the training dataset to the first-principles model. The output of the model is then compared to the actual output and the residuals obtained. These initial residuals are then fed and utilised by the recursive PLS model.

The SVM classifiers are trained using the measured process data without the model parameters or either the non-linearly regressed parameter estimates or the PLS model estimates as specified. When only the kSVM classifier is applied to the datasets the radial basis function kernel is used which is the same SVM configuration utilised by the kernel SVM hybrid fault identification models. This is done to ensure that all model performances are comparable.

The fault detection techniques are applied to identify and differentiate between two fault cases simultaneously. The fault case pairs considered are catalyst deactivation fault - inlet quality fault, and catalyst deactivation fault - heat transfer fault. The application of the fault identification method to the available process data for both the offline and online components are summarised in the steps below:

Offline:

1. Process data is obtained from the CSTR Simulink model for a particular operation mode
2. The process data is labelled based on when faults are implemented
3. Parameter estimates are obtained from the process data using the first-principle model and non-linear regression
4. The process data is centred and scaled and used to train the data-based model
5. The data-based model is used to predict model parameters based on the training data
6. The pairwise SVM classifiers are trained using the process data, parameter estimates and labels

Online:

1. Incoming process data is projected to score variables
2. Parameter estimates are obtained from data-based model based on incoming projected data

3. Process data and parameter estimates are fed to the pairwise SVM classifiers
4. The label for each pairwise classifier is recorded and counted
5. The label for a specific observation is set as the majority of labels obtained from the pairwise classifiers

Once the training process data is available it is subjected to k-fold cross validation where 10 folds are considered. The hybrid models are trained using the training datasets following the procedures outlined above. The performance of the hybrid models is evaluated using the test set by comparing the model parameter predictions made using the models to the true values of the models. The hybrid models are then incorporated with SVM classifiers to form the fault identification method described in Section 3. The performance of the fault identification methods is quantified and compared using the sensitivity and specificity measures.

The MATLLAB scripts and function file used to implement the CSTR model, fault implementation, parameter regression, PLS models and fault identification have been uploaded to GitHub. These files represent the code used to generate the results discussed in this study. The GitHub repository can be accessed at the following link: https://github.com/Stellenbosch-University-Process-Eng/Hybrid-modelling-feature-extraction-for-fault-identification

5 RESULTS AND DISCUSSION

The performance of the fault classification methods for the different process faults and fault identification models are outlined and discussed in this chapter. Firstly, the effects of the process faults on the system are evaluated and discussed. The performance of the standard and kernel SVM's as fault identifiers are evaluated and compared. The performance of the classifiers using the non-linear regression estimates of the unknown process parameters (k and UA) are compared and evaluated to determine if utilising the information about the process parameters will increase the classifier performance. The hybrid modelling based fault identification models are tested and the performance quantified and compared when the different hybrid models and data-based models are used. The values of the sensitivities and specificities for each cross-validation test case is included in Appendix B. The values included in the sensitivity and specificity curves are given in Appendix B as well.

5.1 Effects of process faults

The process faults are applied to process variables and parameters which are not directly measured and recorded in the process. The process faults do however influence the process output variables namely the outlet temperature and concentration. The effect of the process faults on the process output variables are compared in Figure 21 and Figure 22 below. The catalyst deactivation fault has a clear effect on the outlet temperature thus it is expected that the fault identification procedure will perform well when this fault is active. The inlet quality fault also appears to influence the temperature resulting in a gradual increase. It also noted that the effects of inlet concentration fault are still seen after the fault is no longer actively implemented with the temperature still increasing. The effect of the heat transfer coefficient fault on the temperature is seen with the temperature values gradually increasing while the fault is active. When the outlet concentration values are compared, the inlet concentration fault does appear to influence the concentration with a decline in the concentration noticed when the fault is active. The other faults do not appear to have any significant visible effect on the outlet concentration over the time span displayed.

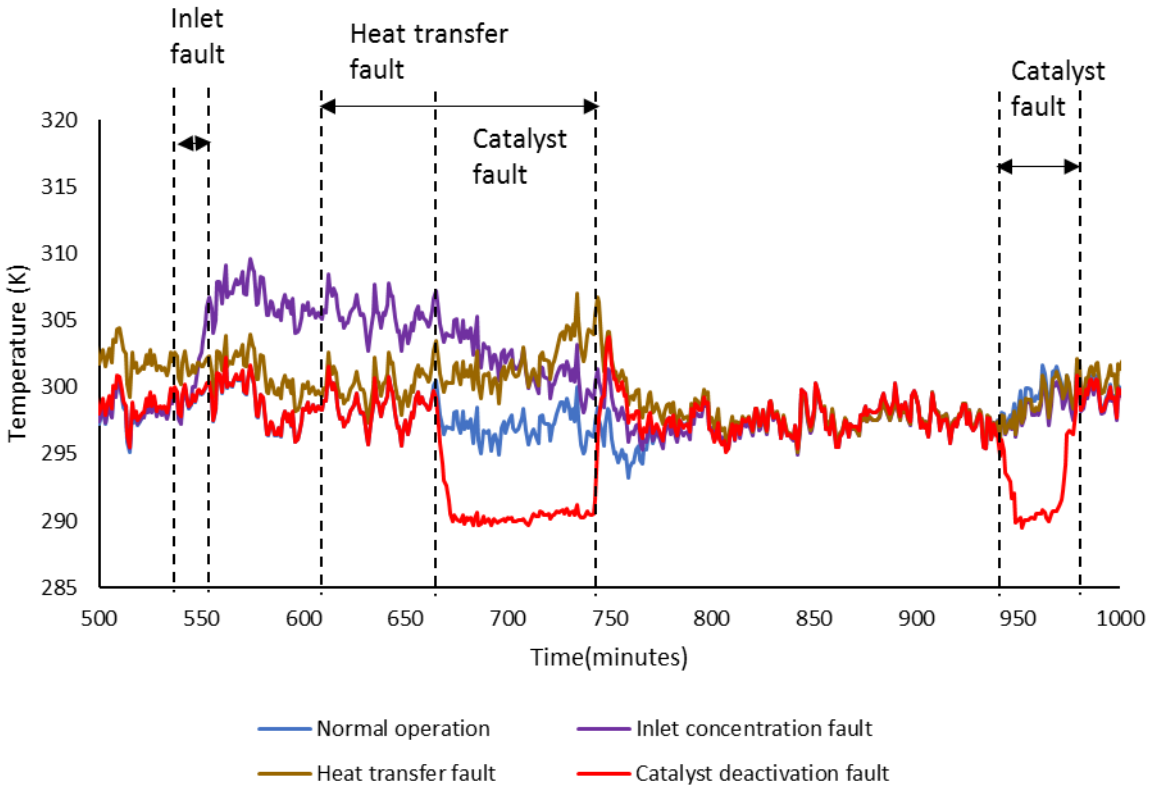

Figure 21: Outlet temperature under different operating conditions

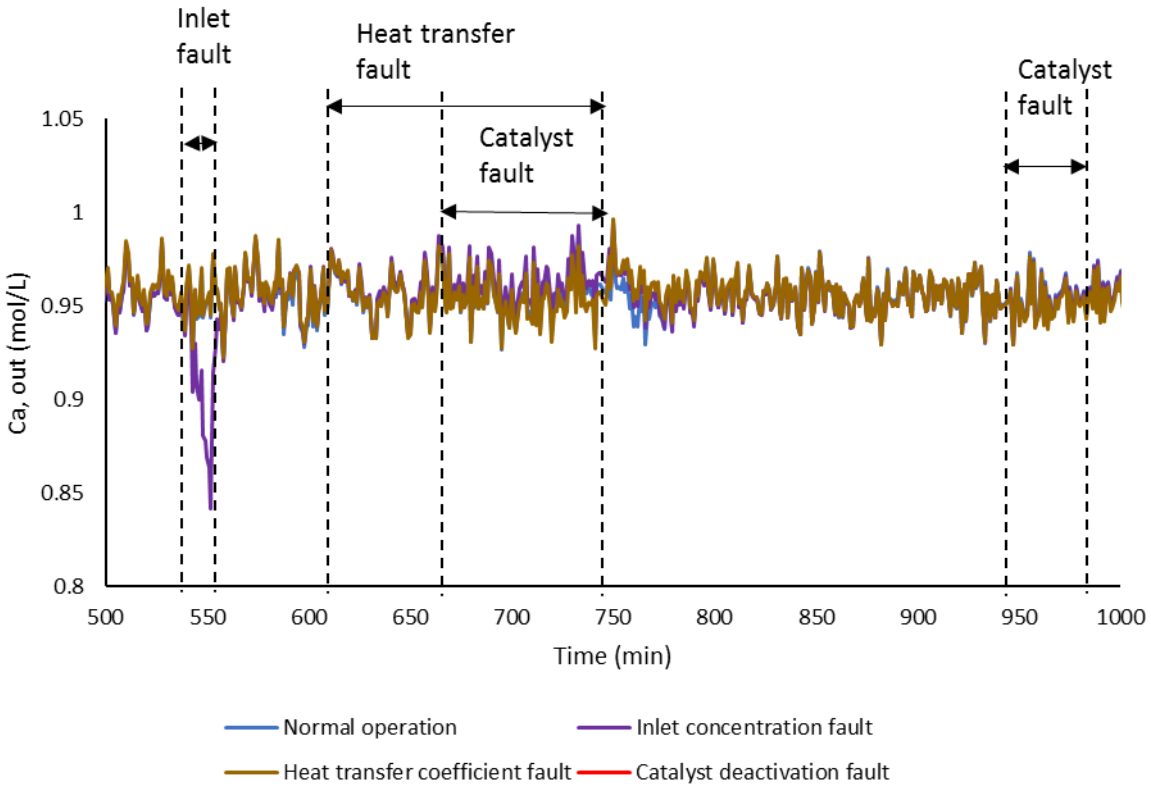

Figure 22: Outlet concentration under different operating conditions

A PCA is performed on the generated datasets to visualise the data and to examine whether the process fault laden observations form groupings which can be identified through the PCA. The first two principal components are retained as these capture the majority of the variance of the datasets. This is illustrated in Figure 23 below where a catalyst deactivation fault laden dataset with a fault magnitude of 62.5%.

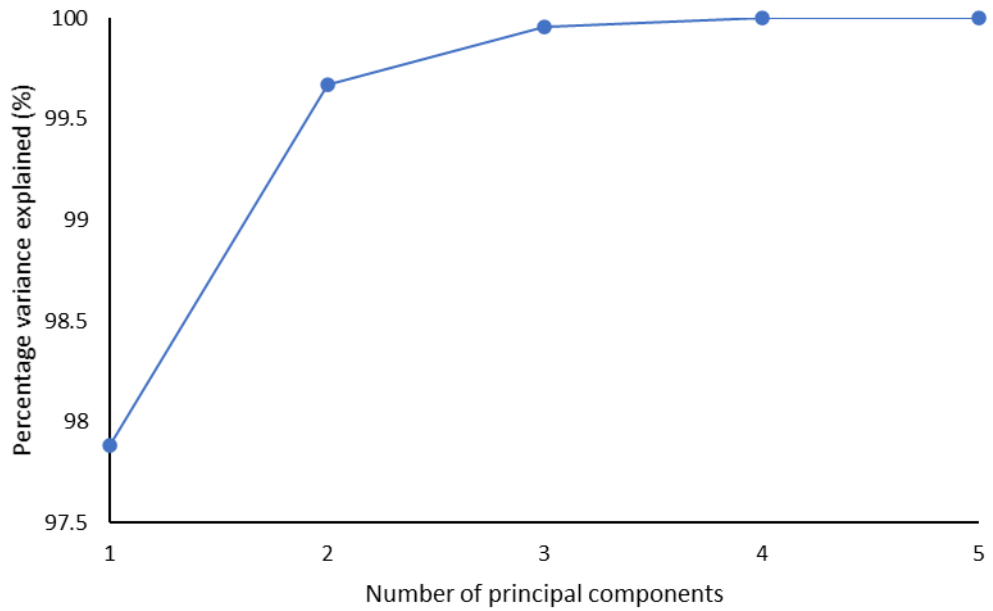

Figure 23: Cumulative percentage variance explained by principal components for catalyst deactivation fault dataset

The first two scores of each observation are taken and illustrated on the plot shown in Figure 24 below. Two components are selected as for the datasets generated in the study the bulk of the variance is captured in the first two principal components. On the plot it is seen that the catalyst deactivation fault observations cluster together and form groupings. This is promising as if the faults form identifiable groupings a classifier (in this work a SVM or kSVM classifier) should be able to identify observations belonging to the respective fault. The heat transfer fault observation does form a group however it overlays with the normal operation observations and is thus not discernible from the normal operation observations. The fault identification models thus might not be able to correctly identify the fault. The inlet concentration faut does not form any visible groupings and thus raises some concern on whether the fault would be accurately detected by the fault identifier. This could be due to the loss of information which by only showing two of the principal components with the additional components possibly

containing information which could result in a more identifiable grouping. Including estimates of unknown process parameters could also result in a clear grouping of the inlet concentration fault and heat transfer fault.

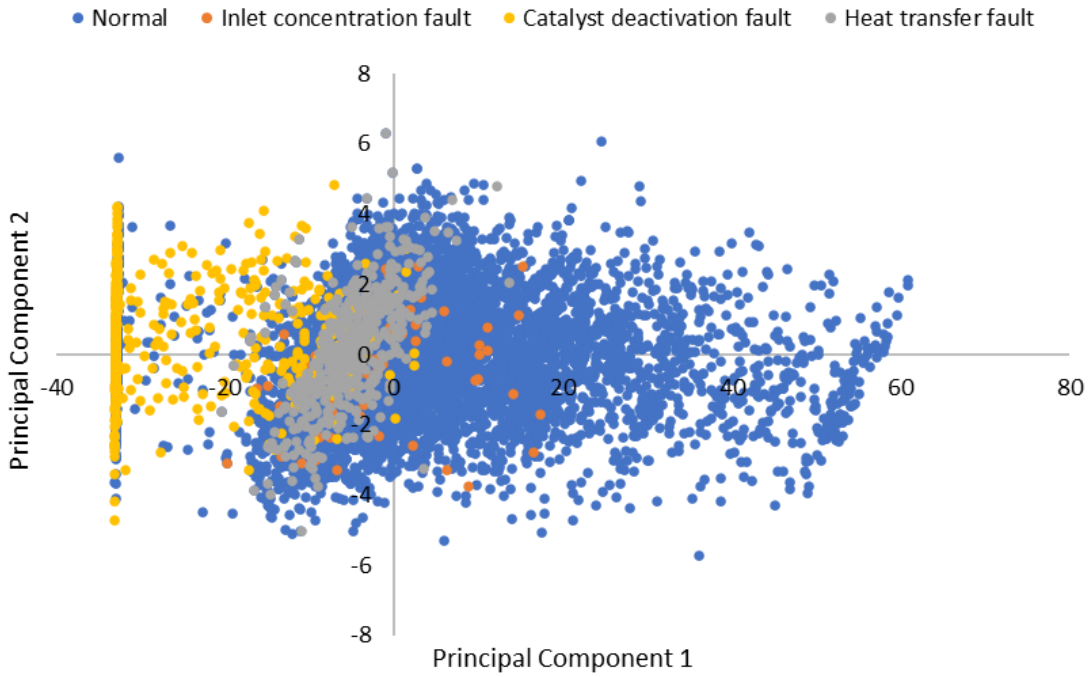

Figure 24: Plot showing first 2 principal components for each fault case

5.2 Standard SVM and kernel SVM performance

As the SVM classifier used in this work has the capability to make use of kernel transformations one should determine if the use of kernels is necessary. A preliminary evaluation of the performance of the SVM and kernel SVM (kSVM) models is done by using receiver operating characteristic (ROC) curves. The RBF kernel with the width parameter set at one was used for the comparison. The ROC curves compare the true positive rate (sensitivity) and the false positivity rate (1-specificity) as a function of the posterior probability threshold. The threshold value is varied incrementally from 0 to 1 and the true and false positivity rates are recorded and displayed on the curve. This provides an indication of the performance over the entire range of probabilities. SVM classifiers does not make use of posterior probabilities thus Platt scaling is used (Platt, 2000). The fitPosterior function in MATLAB was used to fit posterior probabilities to the classifiers through Platt scaling. The

classifiers were trained using a small subset of data containing normal operation data and catalyst deactivation fault data. The ROC curves are shown in Figure 25 below.

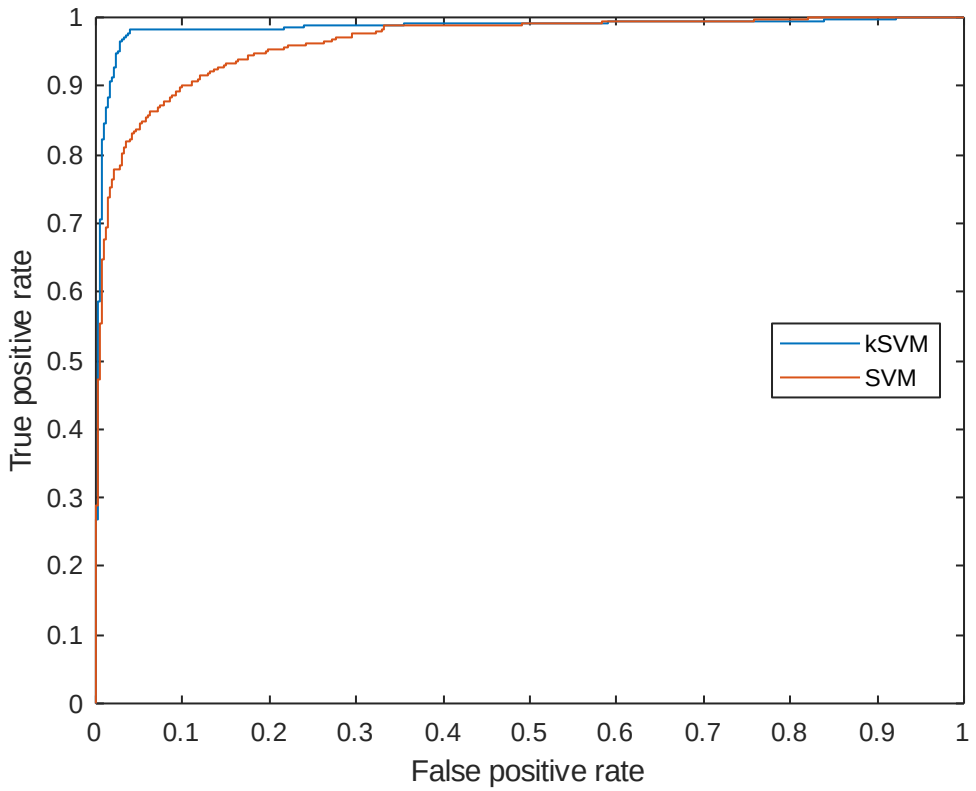

Figure 25: ROC curves for kSVM and SVM classifiers for catalyst deactivation fault

The performance of the classifiers is compared by evaluating the area under the curve (AUC) for the ROC curves generated for each classifier. When the AUC for each curve was determined it was found that the AUC for the kSVM was 0.9865 and the AUC for the standard SVM was 0.9650. As seen the kSVM does show a slight increase in performance when compared to the SVM. Based on this increase in performance the kSVM using a RBF kernel with a width parameter of one was used in this study. The slight increase however does not provide enough evidence to conclusively state that the kernel SVM will outperform the standard SVM on all fault cases considered in this work. The kernel SVM can be more computational resource intensive due to the use of kernel transformations thus the slight increase in performance does not justify the exclusion of the standard SVM classifier. Thus, further evaluations on the performance of both classifiers will need to be conducted. The fitting of posterior probabilities to results of the SVM classifiers was only done for a preliminary comparison of the SVM and kSVM performance. The performance of the fault identification models is evaluated using the

outputs of the SVM models without posterior probabilities fitted to the trained model outputs (thus showing the fault identification performance based on the classes assigned by the SVM models).

As described in Chapter 4 the performance of the classification method is quantified and compared using the sensitivity and specificity measures. The fault identification method is evaluated on each available cross-validation test case with the sensitivities and specificities averaged and compared. The error bars included are based on the standard deviation for the sensitivity and specificity amongst the test case values. The sensitivities and specificities of the SVM classifiers when applied to the catalyst deactivation-inlet concentration fault pair where the catalyst deactivation fault is active are compared in Figure 26.

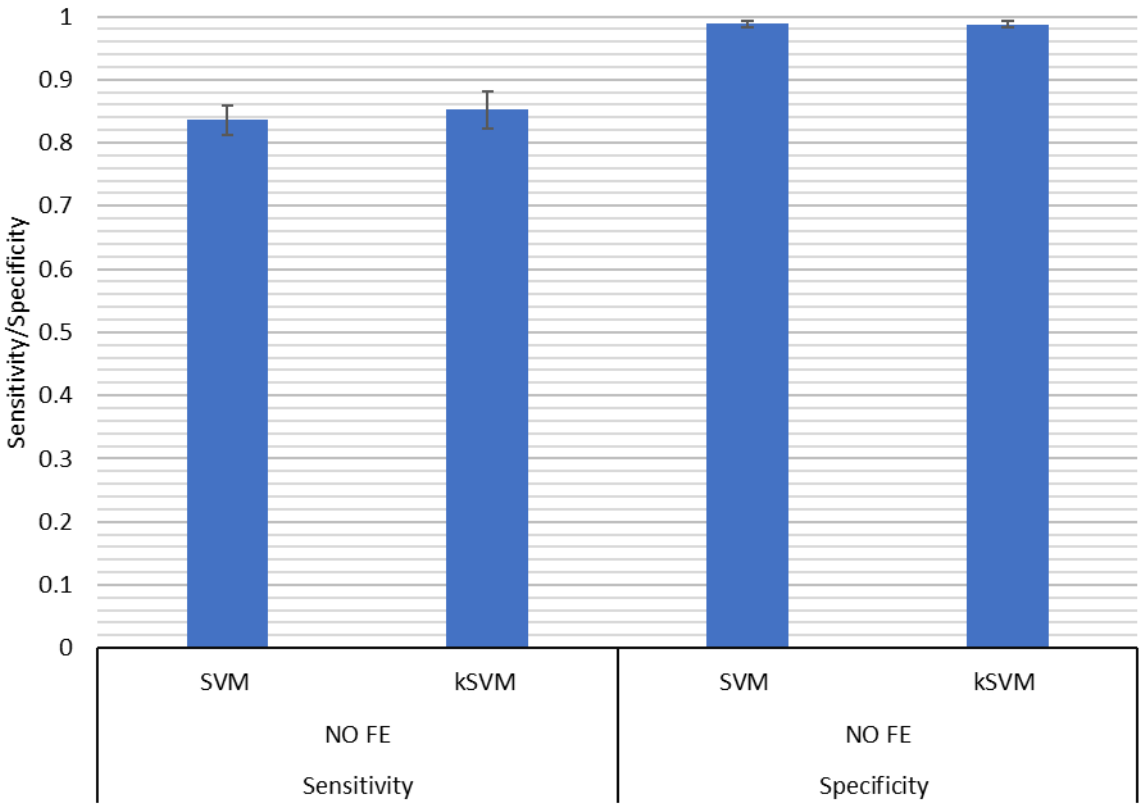

Figure 26: Sensitivity and specificity for standard and kernel SVM fault identification models with no FE (feature extraction) when catalyst deactivation fault is active

The sensitivities represent the portion faulty datapoints which are correctly identified as such, and the specificity represents the portion of non-faulty cases correctly identified as such. The

kernel SVM shows a higher sensitivity with a value of 0.852±0.030 than the standard SVM models which shows a value of 0.836±0.023. This indicates that by implementing the kernel transformation the classifier's ability to correctly identify when the system is at fault is improved. This corresponds with the results obtained from the ROC analysis. Thus, the model correctly performs its function as both a fault detection and fault identification model as it can correctly distinguish between normal and faulty operating conditions. It is seen that both models have a high specificity rate with 0.988±0.005 for the SVM and for 0.988±0.005 the kSVM which indicates that both models result in a low false alarm rate for both models.

When the fault detection method was applied to the inlet concentration fault, the model was not able to identify the fault at the listed fault magnitudes. The inlet concentration fault however does have a visible effect on both the outlet concentration and temperature. The non-identification of the inlet concentration fault is likely due to the labelling procedure followed in this study where the observations where the fault is active are labelled as fault laden.

The inlet concentration fault effects however are still felt once the fault is no longer active, resulting in observations where the effect of the fault is experienced being labelled as normal operation. This results in fault identifier not being able to differentiate between normal operation and the inlet concentration fault. A solution to this is by following a more traditional fault identification method whereby a separate fault detection model is first applied. Once the fault is detected it is then subjected to fault identification, this would remove the issue of labelling the data at the time instance at which the fault is active. Another solution is altering the data labelling process followed, where a fault detection model is applied to the training data where the fault is active. The data entries which are identified as faulty can then be attributed to the fault active in the considered training dataset (i.e. if a fault is detected in the catalyst deactivation training set the fault is considered as a catalyst deactivation fault).

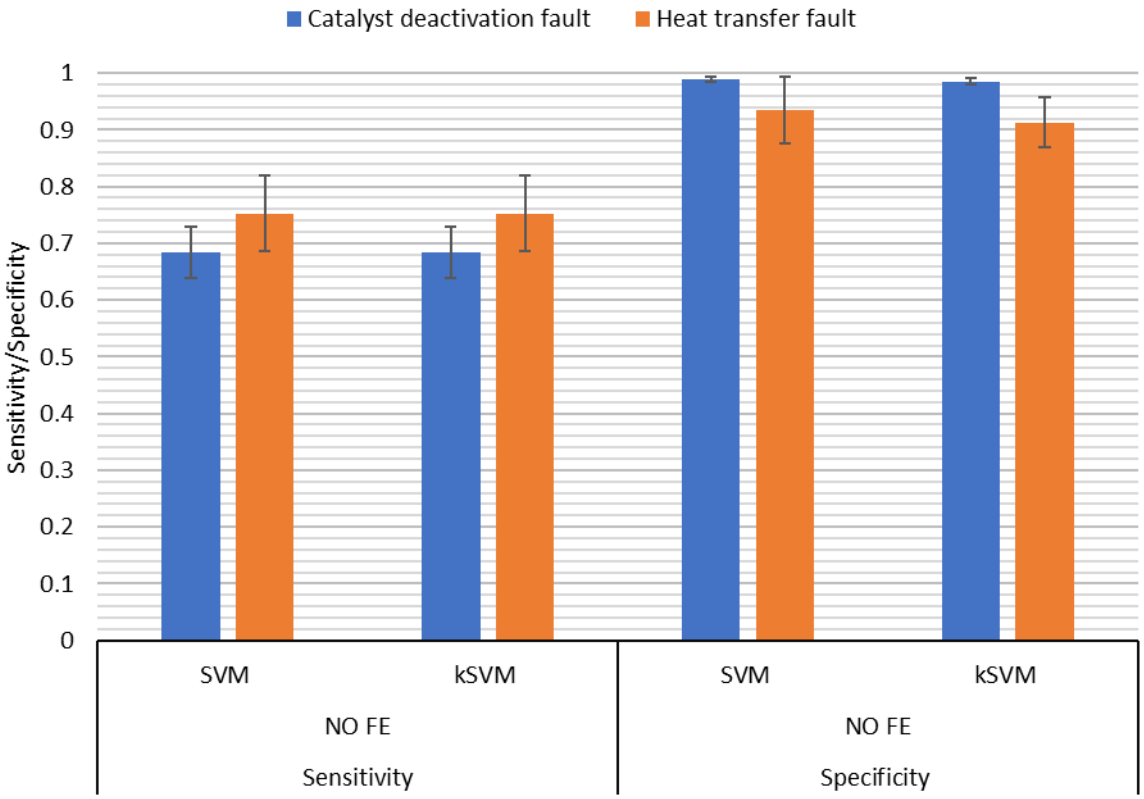

Figure 27: Sensitivity and specificity for SVM and kSVM applied to catalyst deactivation – heat transfer fault pair

The ability of the fault identification method to discern between two process faults still needs to be evaluated. The ability of the SVM classifiers to differentiate between the catalyst deactivation and heat transfer fault is evaluated by applying the classifiers to the concerned fault pair. The sensitivity and specificity measures obtained when the standard and kernel SVM are applied to the fault pair are shown in Figure 27 above. The sensitivities displayed for the catalyst deactivation fault are lower than those seen in Figure 26 due to the model being trained on a different dataset.

The results presented in the two figures above represent the baseline performances to which the other fault identification models can be compared. The sensitivities shown by the kSVM and SVM were found to be identical for both faults with a value of 0.684±0.044 for the catalyst deactivation fault and 0.752±0.067 for the heat transfer fault. The classifiers display a higher detection rate (shown by a higher sensitivity) when identifying the heat transfer fault when compared to the catalyst deactivation fault. When the specificities are examined the SVM models do show slightly higher specificities for both the catalyst deactivation fault

(0.989±0.005) and heat transfer fault (0.934±0.058). The kSVM model specificities however do not show a large decrease for both the catalyst deactivation (0.989±0.005) and heat transfer fault (0.934±0.058). To further evaluate the performance of the SVM classifiers the training time of the models on the catalyst deactivation-inlet concentration fault training set are compared.

The training and execution time of both models were recorded and compared. The kernel SVM fault identification model reported a total training time of 624 seconds (or 10 minutes 24 seconds) while the standard SVM fault identification model reported a total time of 30053 seconds (or 500 minutes 53 seconds). The increase in execution time of the standard SVM fault identifier could be attributed to the increase in training time of the SVM model. This could be due to the SVM taking a longer time to find an optimal position for the separation hyperplane which sufficiently separates the classes. This is likely due to the data set being non-linear. The kernel in the kernel SVM transforms the dataset to a space where the data possibly becomes linearly separable. This decreases the time taken to find an optimal position of the separating hyperplane and identification of the support vectors. This further indicates that the kSVM is a viable choice for the form of the SVM classifier used in the fault identification model.

5.3 Fault identification models using non-linearly estimated parameters

To evaluate whether introducing estimated parameters to the fault identification models improves the performance of the models, non-linearly regressed parameters are fed to the classification models. The non-linearly regressed parameter estimates represent the ideal estimates for the unknown process parameters however performing the non-linear regression in an online application is not feasible due to the execution time of the regression. The performance of the non-linear regression is evaluated by constructing parity plots for both the reaction rate and grouped heat transfer coefficient estimates when the respective faults are active as seen in Figure 28 and Figure 29 below.

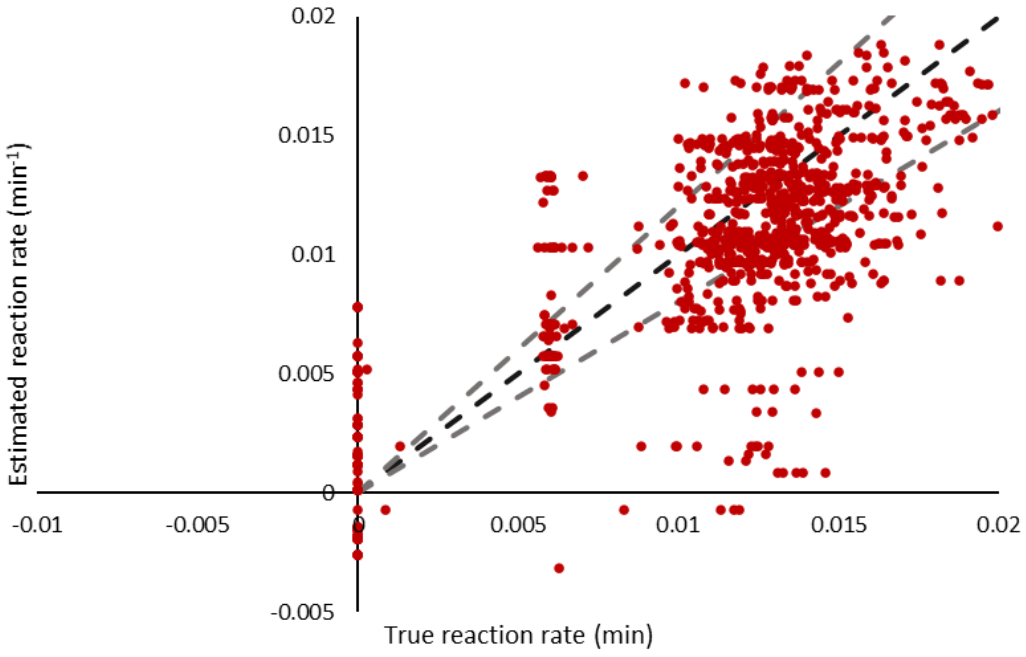

Figure 28: Parity plot for non-linear regression reaction rate estimates where catalyst deactivation fault is active with 20% relative error margin (grey dashed line) and 45° line (dark grey dashed line)

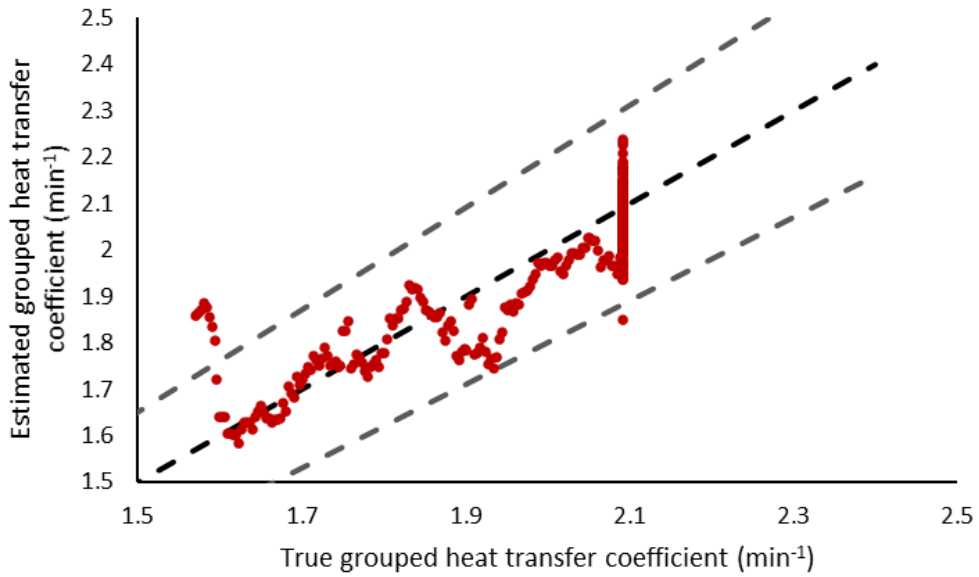

Figure 29: Parity plot for non-linear regression heat transfer coefficient estimates where heat transfer fault is active with 10% relative error margin (grey dashed line) and 45° line (dark grey dashed line)

When model performance is evaluated using parity plots, the better performing the model the closer the data points lie to the 45° line on the lie. In Figure 28 it is seen that there are entries which lie on the y-axis. These represents the entries at which the catalyst deactivation fault is active, and the reaction rate drops to 0 min^{-1}. The poor prediction of these observations results in the spread of the entries along the y-axis with the ideal predictions lying close to the origins. As seen in Figure 28 there is noticeable scatter surrounding the 45° line with many of the data entries lying outside the 20% relative error margin.

In Figure 29 the data entries display less scatter, and the majority of the entries lie within 10% margin. This indicates that the non-linear regression estimates for the heat transfer coefficient when the heat transfer fault is active are more accurate than the estimates for the reaction rate when the catalyst deactivation fault is active. One would expect the fault identifier to perform better when identifying the heat transfer fault when the classifier is fed the parameter estimates. As the changes in the parameter are accurately captured by the estimated parameters this suggests that changes due to the fault are captured. By capturing the change in the heat transfer coefficient value due to the fault it should allow the fault to be more easily identified.

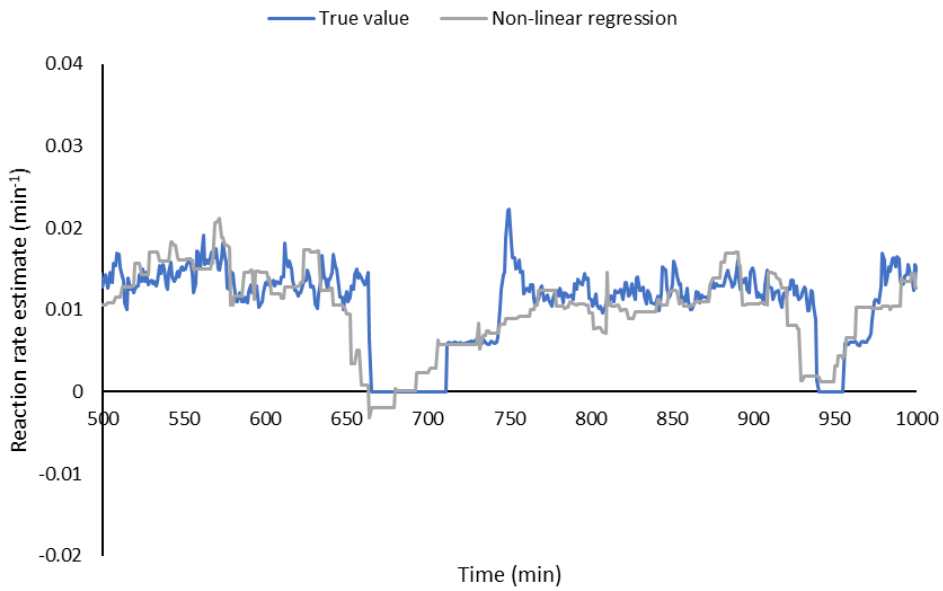

Figure 30: Time series plot of non-linearly regressed reaction rate estimates when catalyst deactivation fault is active

The non-linear regression can further be evaluated by comparing the non-linear regression estimates to the true values using a time series plot. The reaction rate estimates subject to the catalyst deactivation fault and the heat transfer coefficient subject to the heat transfer fault are shown in Figure 30 above and Figure 31 below respectively.

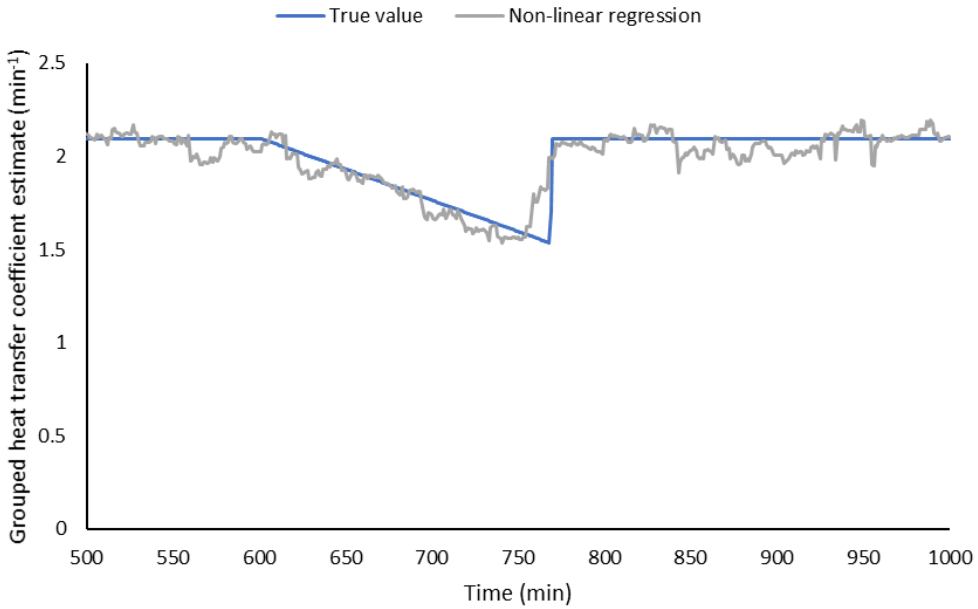

Figure 31: Time series plot of non-linearly regressed grouped heat transfer coefficient estimates where heat transfer fault is active

As seen in the plots above the parameter estimates of both the reaction rate and the heat transfer coefficient are able to track changes in the parameters due to the effects of the fault. It appears that the reaction rate estimates do closely track the true values which is not suggested or evident by the parity plot. This discrepancy between the expected performance and actual performance is likely due to the small magnitude of the reaction rate. A slight difference between the predicted and true values will lead to a large error. The entries on the parity plot which show the largest deviation from the ideal performance corresponds to observations where the catalyst deactivation fault is active. For these observations the non-linear regression either overshoots or undershoots the magnitude of the decline. It is noted that the negative reaction rate which is sometimes estimated by the non-linear regression is unrealistic and is a result of the regression technique used.

The error of the regression technique is quantified using a scaled mean absolute percentage error (MAPE) measure as defined in Equation 50 below. When the MAPE was calculated for

the fault cases it was found that the MAPE value for the catalyst deactivation fault was 26.9% and the MAPE for the heat transfer fault was 2.72%. This coincides with the parity plots which illustrates that the heat transfer coefficient regression when the fault is active is more accurate than the reaction rate estimates when the catalyst deactivation fault is active. As the scaled MAPE value makes use of the mean of the true values, when the catalyst deactivation fault is active small differences between the true and predicted values leads to large MAPE values.

$$MAPE = mean\left(\frac{|Predicted - True|}{mean(True)}\right) \quad [\ 50\ ]$$

To identify whether the parameter estimates will aid in separating and grouping the different faults a parameter comparison plot is generated where reaction rate estimates lie on the x-axis and heat transfer coefficient estimates on the y-axis. The plot allows one to evaluate whether there are any distinctions between the fault conditions with respect to the parameter estimates. The plot is shown in Figure 32 below where the parameter estimates were scaled to ensure the parameters are of comparable magnitude. The scaling is performed by dividing the parameters by their NOC values and then multiplying the value by 0.9. This results in the NOC value being assigned the value of 0.9 allowing for values to overshoot the NOC value while trying to maintain a maximum value of 1 for the scaled parameters. By using scaled parameters, one still can get a sense of whether groupings form however the actual magnitudes of the parameters might skew the results.

When examined it is seen that heat transfer fault observations form a concentrated cluster however the cluster overlays the normal operation data thus not being distinct from the normal operation data. Based on the following plot one cannot determine whether the parameter estimates will improve the identification of the heat transfer fault. The catalyst deactivation fault does not form as close a cluster as the heat transfer fault but does still form a grouping. A notable point is that there are visible catalyst deactivation data entries which are separate from the normal operation observations. This could aid in achieving efficient classification of the catalyst deactivation fault. The inlet concentration fault does not form as close a grouping as the heat transfer fault. The inlet concentration fault observations show some scatter and the entries do overlap with the normal observation entries. Thus, it is not clear as to whether the parameter estimates will improve the identification of the fault as the grouping is not as clear

as the other faults. The inlet concentration fault also does not appear to have observations which are separate from the normal operating conditions.

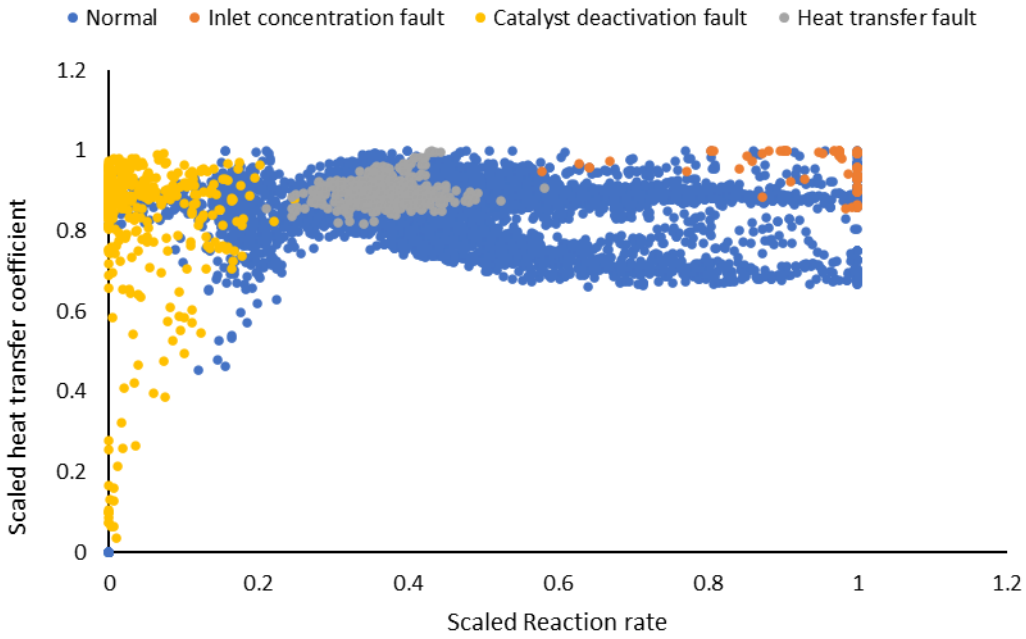

Figure 32: Scaled non-linearly regressed parameter estimates for each fault case

The non-linearly regressed parameter estimates are fed to the fault identification models to evaluate whether the process parameters do impact the performance of the classifiers. The sensitivities and specificities for the catalyst deactivation-inlet fault pair and catalyst deactivation-heat transfer fault pair are shown in Figure 33 and Figure 34 respectively. For the catalyst deactivation-inlet concentration fault pair the sensitivity and specificity of both the r-SVM (0.905±0.026 and 0.996±0.005) and r-kSVM (0.887±0.019 and 0.998±0.003) models show an increase in both sensitivity and specificity. The shows that the process parameters can increase the performance of the classifier. The r-SVM model shows the best performance with respect to considered fault pair.

When the r-SVM and r-kSVM are applied to the inlet concentration fault test case the fault still cannot be detected with the additional parameters fed to the classification models. Based on this test case it can be concluded that if the SVM classifier itself cannot detect the fault, then implementing the hybrid modelling techniques will not result in the fault identification method being able to detect and identify the fault. It can also be stated that if a fault is

implemented on a process parameter which is neither measured nor estimated, the identification technique is not able to identify the process fault.

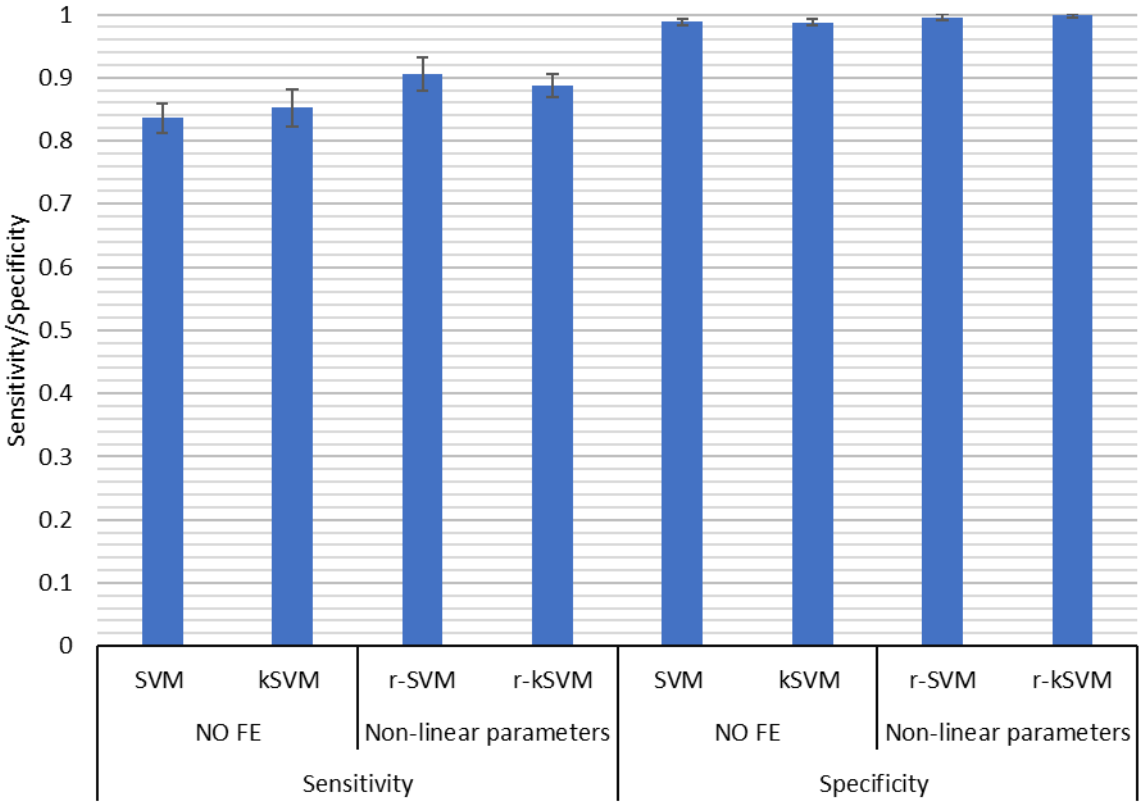

Figure 33: Sensitivity and specificity for r-SVM and r-kSVM for catalyst deactivation-inlet fault pair with catalyst deactivation fault active

When the parameters are fed to the catalyst deactivation-heat transfer fault identification models an increase in performance is experienced. The sensitivity of the r-SVM models to detect both the catalyst deactivation (0.686±0.042) and heat transfer fault (0.811±0.032) increases noticeably. This is due to the parameter estimates tracking the changes in the parameters due to the effects of the fault. There is no significant in the specificity of identifying the catalyst deactivation fault (0.989±0.005) r-SVM model with an increase in the specificity when detecting the heat transfer fault (0.968±0.027). The increase in specificity is likely the result of the additional parameters creating clear separation boundaries between the classes. This in turn results in less false fault identifications resulting in a higher specificity value.

The sensitivity and specificity values for the r-kSVM model when identifying the catalyst deactivation fault identification are 0.633±0.058 and 0.974±0.005. The results for the heat transfer fault identification are 0.734±0.033 and 0.924±0.038. The r-kSVM model does not

display an increase in the performance of the classification model but rather shows a decrease in performance. The decrease is due to the parameter estimates causing overfitting of the SVM model thus decreasing the performance.

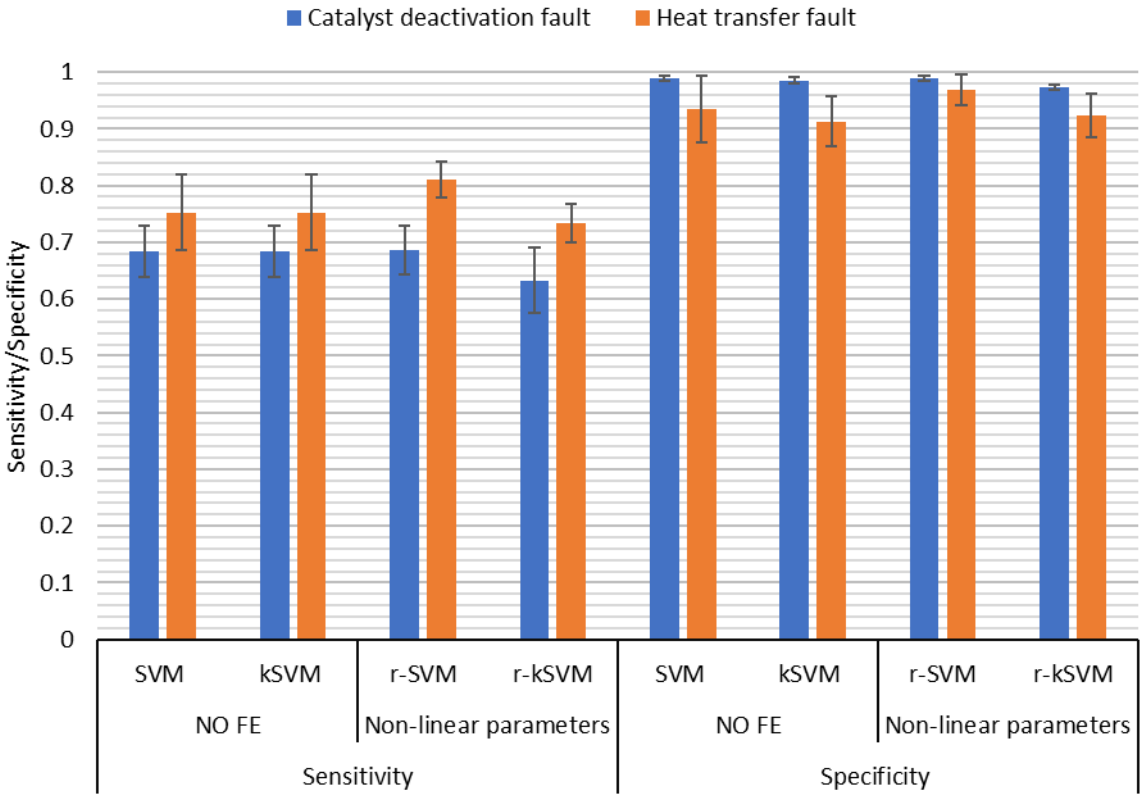

Figure 34: Sensitivities and specificities for r-SVM and r-kSVM applied to catalyst deactivation – heat transfer fault pair

5.4 Fault identification models using shortcutting hybrid modelling

The hybrid modelling approaches applied to the fault identification models can be categorised based on the data-based model used namely PLS and dPLS. It was shown that feeding estimated process parameters to the SVM classifiers improves the performance of the fault identification models. The hybrid models are proposed to reduce the execution time of the fault identification models as non-linear regression is time consuming. In the following chapter the performance of the hybrid models which use PLS and dPLS models were compared.

5.4.1 Standard and dynamic PLS

When using PLS regression models an important decision that needs to be made is selecting the number of components used by the regression model. This can be done by considering the cumulative sum of the variance accounted for by each PLS component. This provides an indication as to how well the PLS model describes the original training dataset as when applying PLS models not all components are required for the regression. A standard PLS model was trained using a dataset containing catalyst deactivation fault data where the fault magnitude is 62.5%. The cumulative variance plot is shown in Figure 35 below. The cumulative variance shown in the figure is based on one case of the 10 possible cross-validation test cases. Most of the variance is explained by the first two components with only a slight increase in variance visible beyond the third component thus only two components are used in the regression.

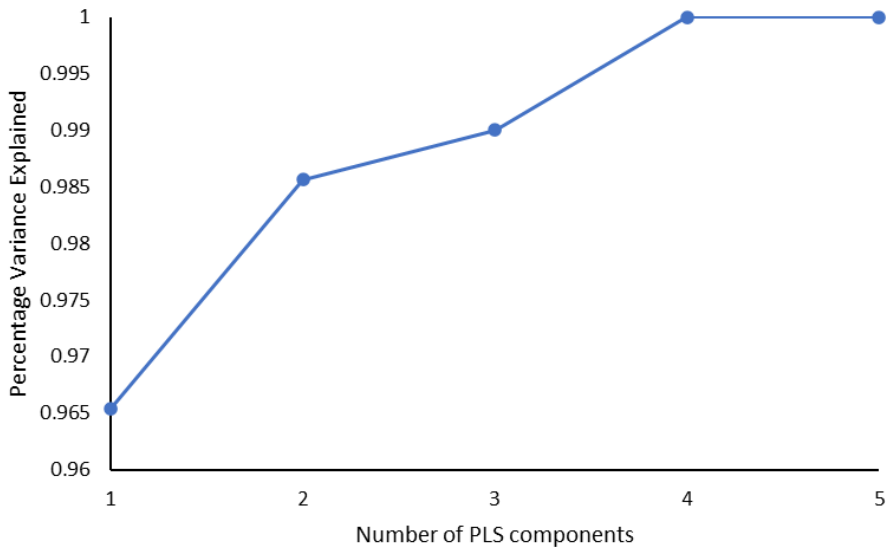

Figure 35: Cumulative variance explained by standard PLS components

The dynamic PLS (dPLS) model can have different time window widths, with the window of the PLS model having a significant effect of the performance of the regression. One wants to select a window width which provides an increase in the accuracy of the regression predictions but not a window so large that it significantly increases both the training time and execution time. It is noted that that both the number of components and window length influences the performance of the dPLS model thus when find determining the optimal configuration for the model both parameters would have to be optimised simultaneously. It is seen however that two

PLS are sufficient for account for the bulk of the variance in catalyst deactivation laden dataset (as shown in Figure 35) with this seen with remaining two faults as well. Thus the sliding window width is selected with model utilising two PLS components. A good initial estimate for a time window length is selecting a time length which encompasses an entire dynamic process cycle. In the CSTR model during which the catalyst deactivation fault is active the process time constant is found to be approximately 8 minutes.

The accuracy and performance of the dynamic PLS model is quantified through the MAPE measure. The MAPE values for different time window widths are shown in Figure 36 with the error bars representing the standard deviation obtained from the MAPE values for each test case allocated by the cross validation. When the MAPE values are compared it is seen that there is decrease in the error until a time window of 6 minutes is reached at which point the values begin to plateau before slightly increasing at 10 minutes. Based on the levelling out of the error values and the process time constant being 8 minutes a time window of 8 minutes is selected. By setting the window width as the process time constant the intention is that the window will efficiently capture the process dynamics.

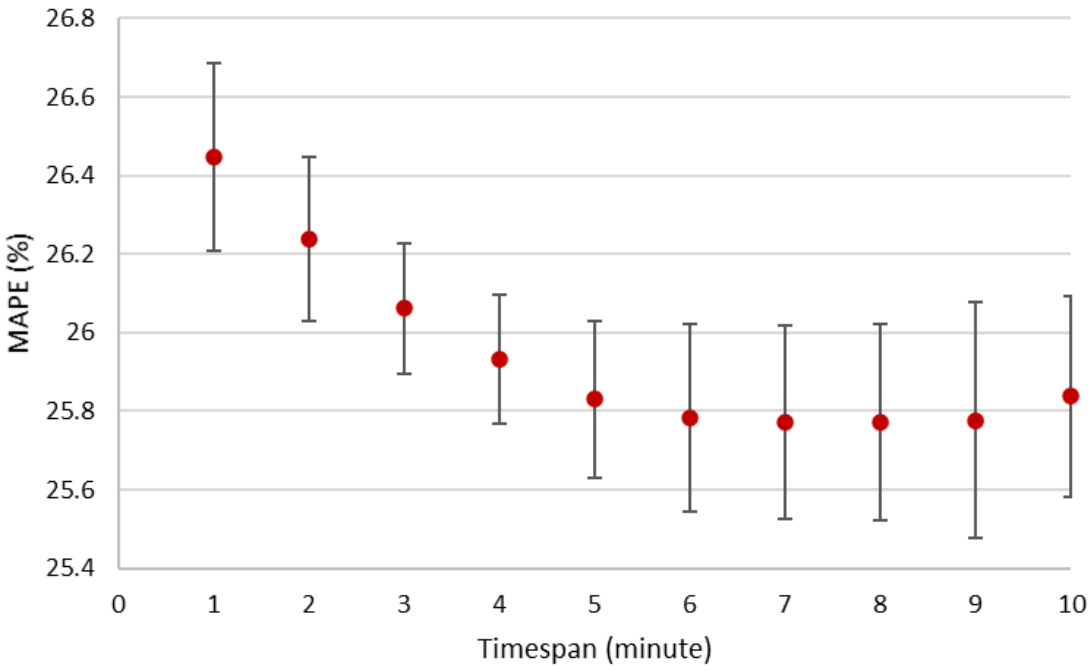

Figure 36: MAPE for different time window widths

In order to evaluate the performance of the various regression methods utilised in the hybrid models, parity plots are generated for each PLS model considered. Parity plots compare the true output of the regression model to the values of the output estimated using the regression model. Parity plots are constructed for the catalyst deactivation fault at specific fault magnitudes. The parity plots for the s PLS model and dPLS model using a catalyst deactivation fault-based test set are shown in Figure 37 and Figure 38 respectively.

As seen in both figures there is considerable scatter around the 45° line with many data entries lying outside the 20% error margin. Many of the data entries lie to the left of the 45° line with these data entries corresponding to observations where the catalyst deactivation fault is active as is evident by the true reaction rate values lie close to zero. The values which are less than zero are due to noise associated with the non-linear regression when the parameter values are obtained during the regression. The parity plots for both PLS models are very similar thus a judgement on the performance of the models cannot be made based on the plots alone.

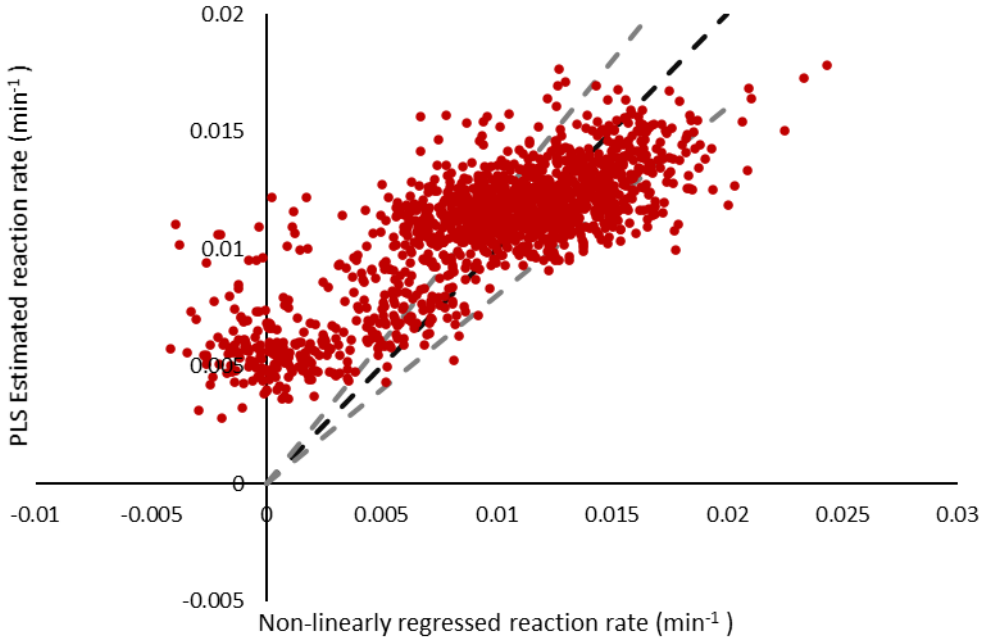

Figure 37: Parity plot for reaction rate using standard PLS model when catalyst deactivation fault is active with 20% relative error margin

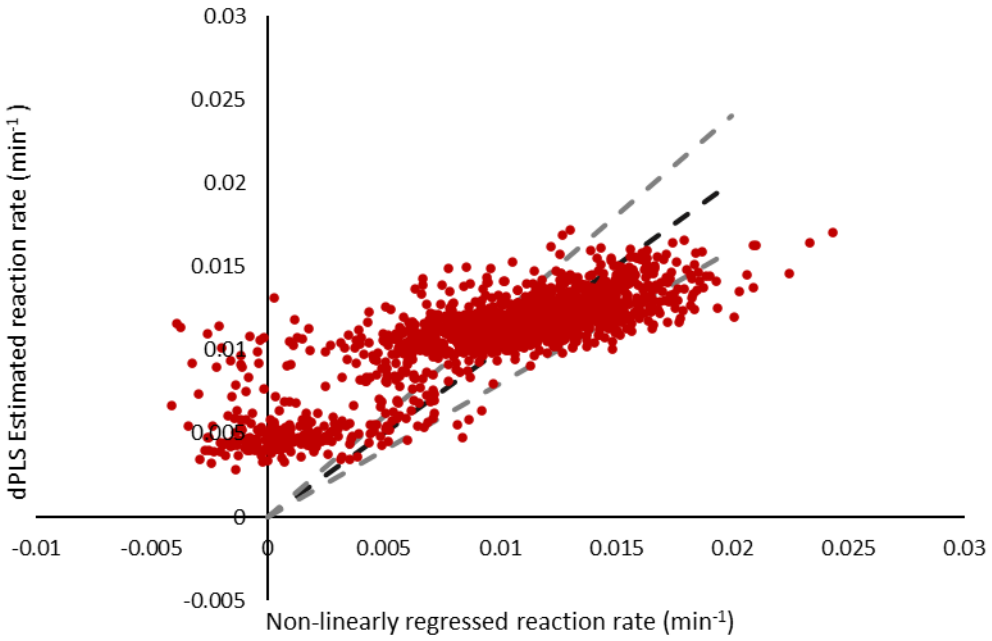

Figure 38: Parity plot for reaction rate using dynamic PLS model when catalyst deactivation fault is active with 20% relative error margin

To further evaluate the performance of the PLS models the MAPE values of the models for the test set illustrated in Figure 37 and Figure 38 are calculated and compared. It is found that the standard PLS model has a MAPE value of 26.66% while the dynamic PLS model shows a MAPE of 25.75%. This indicates that for the test case considered the dynamic PLS provides more accurate reaction rate estimates. To further evaluate the performance of the models a box and whisker plot for the absolute percentage error (APE) is generated and is shown in Figure 39 below.

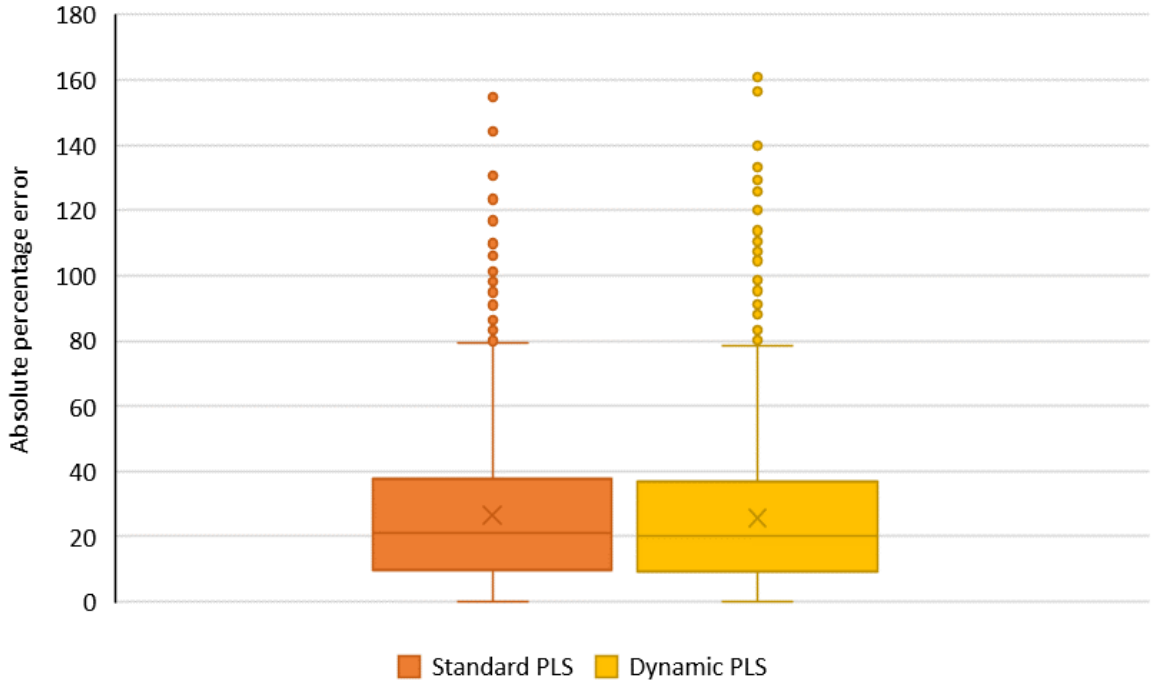

Figure 39: Box and whisker plot comparing APE for standard PLS and dynamic PLS

The box and whisker plot illustrates the variance of the APE values as well as indicating data entries which are outliers. The PLS and dPLS models both display similar variances with regards to the parameter estimates. This is expected as both models performed similarly in their respective parity plots.

To further evaluate the performance of the PLS regression models, the reaction rate is estimated using the trained PLS and dPLS models over the entire time span covered by the simulated process data. The process data is taken from a simulation run when the catalyst deactivation fault active. The estimated reaction rate is compared to the true reaction rate used in the simulation as well as the reaction rate estimated using non-linear regression. The reaction rate estimates are compared in Figure 40.

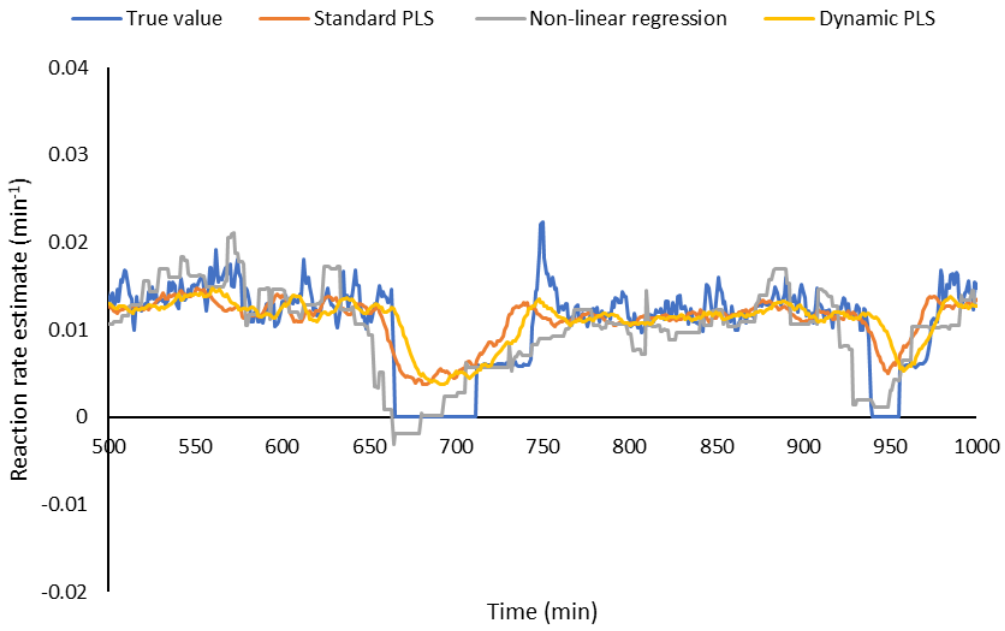

Figure 40: Reaction rate estimates by PLS and dPLS models over a timespan between 500-1000 minutes with catalyst deactivation fault active

As seen in the plot presented the PLS and dPLS reaction rate closely resemble one another. It is seen that the estimates predicted by the PLS models deviate from the true reaction rate and non-linearly regressed reaction rate estimates especially when the catalyst deactivation fault is active. When the reaction rate declines the PLS and dPLS models track and describe the decline in the reaction rate however the models do not match the magnitude of the decline.

In order to further evaluate the performance of the PLS predictions the outlet concentration and temperature are predicted using the parameters obtained from the models. The outlet temperature is predicted and compared to the true temperature readings obtained from the CSTR simulation model in Figure 41 while the outlet concentration is predicted and compared in Figure 42. As seen in both plots there is no significant deviation between the predicted temperature and concentration and those obtained from the simulation. This indicates that although the predicted reaction rate estimates do not fully describe the magnitude of the applied fault this does not significantly affect the outlet temperature and concentration predictions. The MAPE between the predictions and the true simulation values are given in Table 8. Based on the MAPE values given the PLS model shows better performance than the dPLS model.

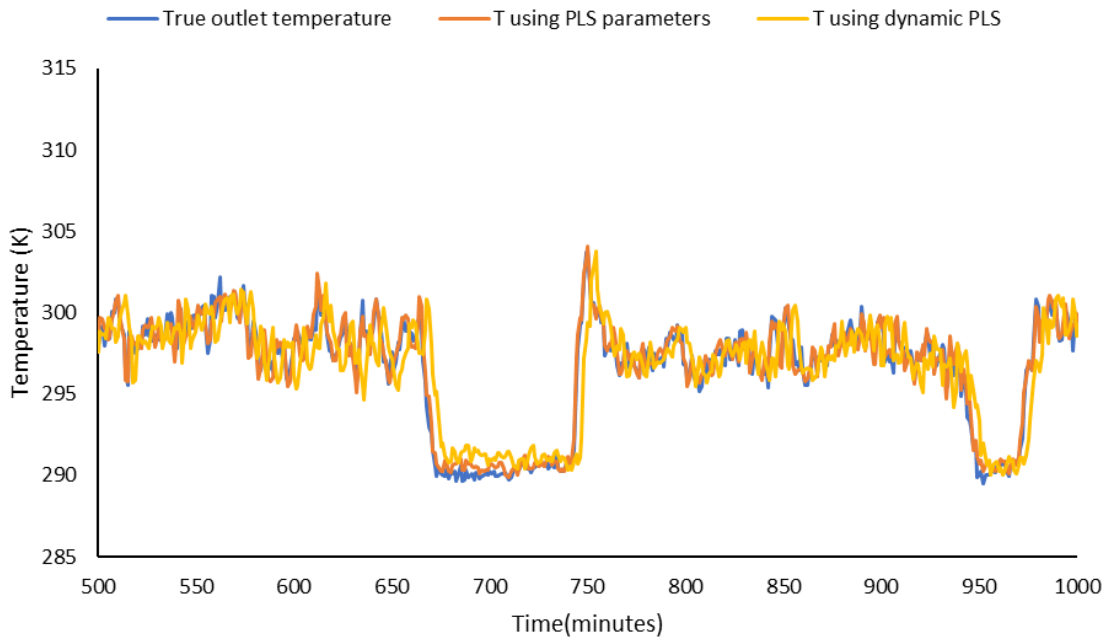

Figure 41: Outlet temperature prediction over a time window of 500-1000 minutes using parameter estimates from PLS and dPLS models

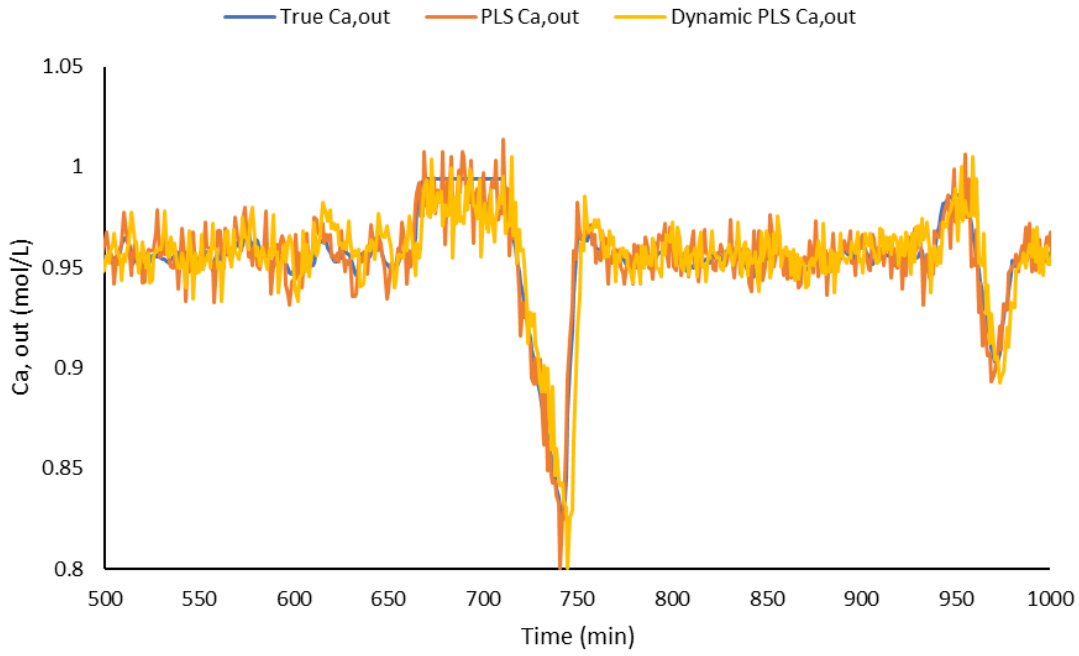

Figure 42: Outlet concentration prediction over a time window of 500-1000 minutes using parameter estimates from PLS and dPLS models

Table 8: MAPE for outlet predictions

Outlet variable	Standard pls	Dynamic pls
Temperature	0.1507	0.4334
Concentration	0.7787	1.017

The performance of the hybrid models when the heat transfer coefficient fault is active is evaluated by predicting the overall heat transfer coefficient using the hybrid model structures considered. The PLS models used in the hybrid models are trained using normal operation, catalyst deactivation fault and heat transfer fault laden data with the training parameters obtained using non-linear regression. The coefficient estimates are compared to the true values as well as those estimated by the non-linear regression in Figure 43 below.

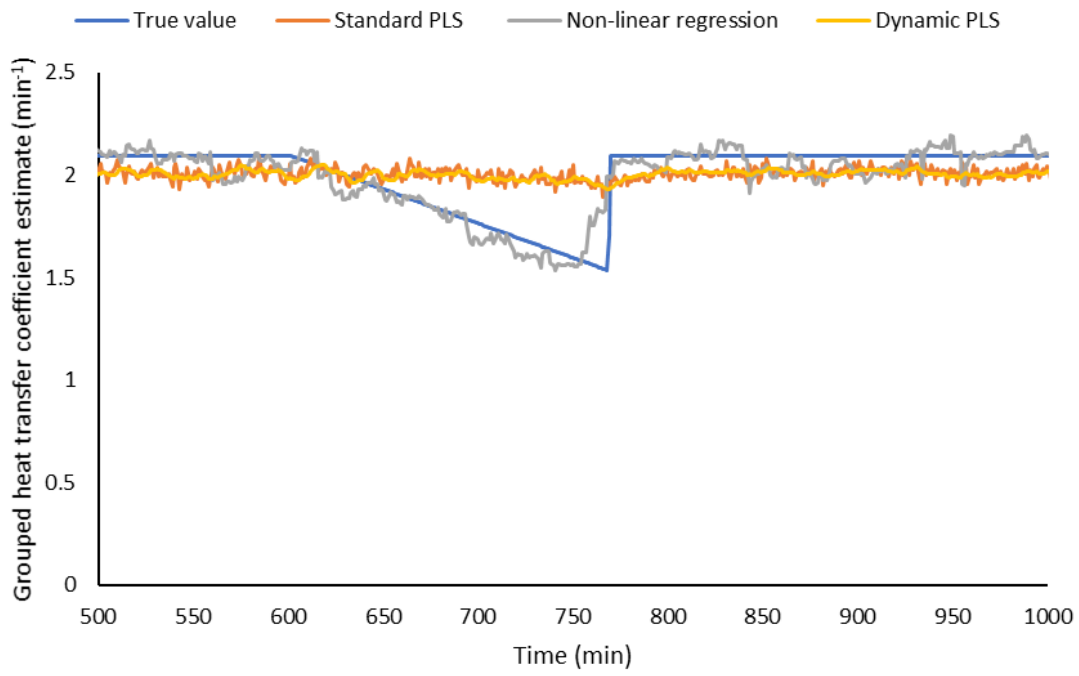

Figure 43: Heat transfer coefficient estimates over a timespan between 500-1000 minutes using parameter estimates from PLS and dPLS models

The true coefficient values are not subjected to noise as the coefficient is based on characteristics of the physical reactor and fluid properties which would not vary. It is seen that when the heat transfer coefficient value decreases due to effect of the fault the estimates

obtained by non-linear regression does appear to track the change in heat transfer coefficient. The scatter visible in the non-linear regression estimates is likely due to the noise on the process variables used in the non-linear regression.

When the predictions of both the PLS and dPLS models are compared to the true values it is seen that the model predictions do not track the decrease in heat transfer coefficient. The PLS models do not display as much variance as the non-linear regression estimates and this could be due to the PLS models being trained on normal operation, catalyst deactivation and heat transfer coefficient fault data. The decrease in variance displayed by the PLS models could also be attributed to the PLS models only utilising two PLS components. It is a common feature of PLS models that the first few PLS components account for the majority of variance displayed in the dataset whereas the latter PLS components commonly account for the noise present in the dataset. The dPLS predictions show less noise on the predictions than the PLS predictions. This is due to the dPLS model considering a set of observations over a time window and not a singular observation. By considering a set of observations over a window the dPLS model minimises the effect of noise on the model regression which results in smoother predictions.

The decrease in the variance displayed by the PLS models could also possibly be due to the variance displayed in the catalyst deactivation fault set mitigating the variance displayed in the heat transfer fault set. When the error between the PLS predictions and the true values is quantified, it is found that the MAPE value for the standard PLS model is 6.17% and the MAPE value for the dynamic PLS is 6.06%. This shows that both PLS models perform similarly when predicting the heat transfer coefficient outputs for the heat transfer fault set.

The reaction rate estimates obtained when the heat transfer fault is active is shown in Figure 44. As expected, the reaction rate estimates obtained using non-linear regression tracks the changes in the reaction rate as exhibited in the true reaction rate values. The change in reaction rate is due to the change in temperature as a result of the heat transfer fault. When the reaction rate estimates are predicted using PLS and dPLS models however it is seen the estimates do not match the changes shown in the true values of the reaction rate. It is seen that the estimates deviate from the true values significantly.

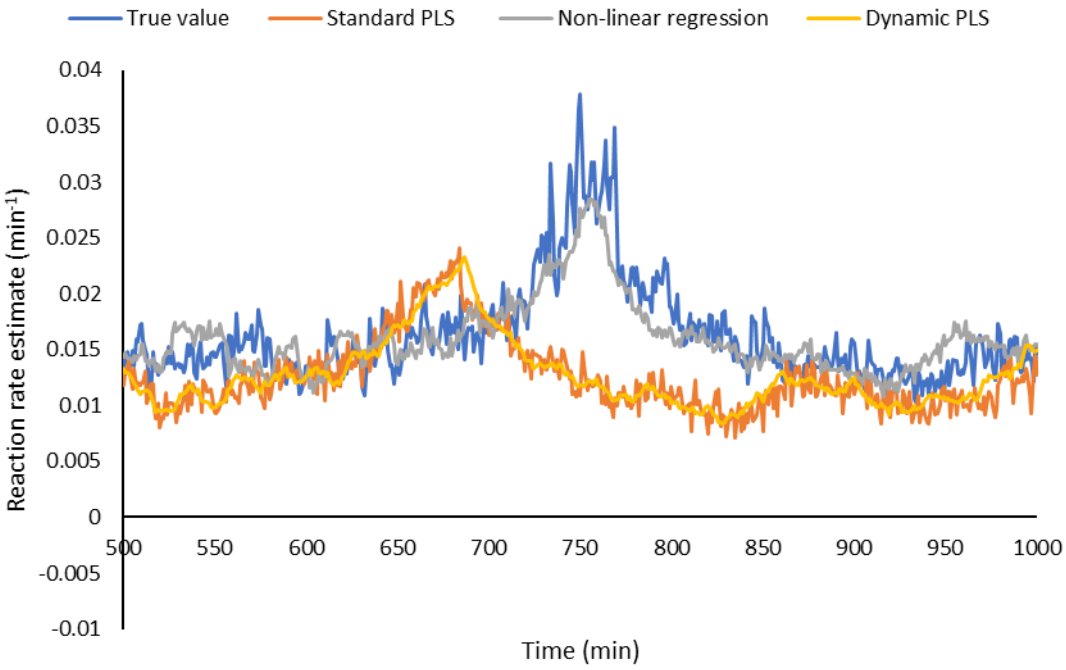

Figure 44: Reaction rate estimates from PLS and dPLS models when heat transfer fault is active

As the heat transfer estimates by the PLS and dPLS model do not show much variation this indicates the PLS models account for the effect of the heat transfer fault through the reaction rate estimates. If the heat transfer coefficient and outlet concentration remain constant the effects of the increase in temperature would be mitigated by the reaction rate remaining constant. As the reaction is exothermic by the reaction rate remaining constant and the heat transfer rate decreasing the temperature increase experienced in the reactor would be due to the heat generated by the reaction. As the rate at which the heat is generated remains constant and the rate at which the heat is removed decreases, the temperature of the reactor would increase.

Parity plots are constructed for the PLS and dPLS models trained using normal operating data, catalyst deactivation fault laden data as well as overall heat transfer coefficient data as shown in Figure *45* and Figure *46*. The performance of the PLS models is compared by considering a test case where the overall heat transfer coefficient fault is active. It is seen in both plots that there exists a noticeable portion of the data entries which lie outside the 10% error bounds. These data entries are due to the predictions at the time instances where the fault is active which indicates that the heat transfer coefficient predictions do not track the changes in the

overall heat transfer coefficient well at all. It is seen that once again both parity plots resemble each other closely, which indicates that both PLS models perform similarly.

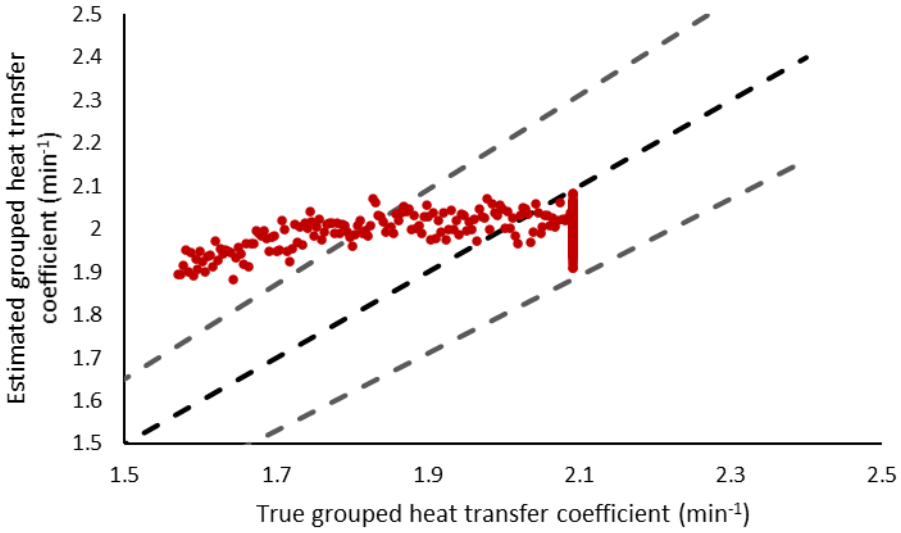

Figure 45: Parity plot for grouped heat transfer coefficient using PLS where heat transfer coefficient error is active with 10% relative error margins

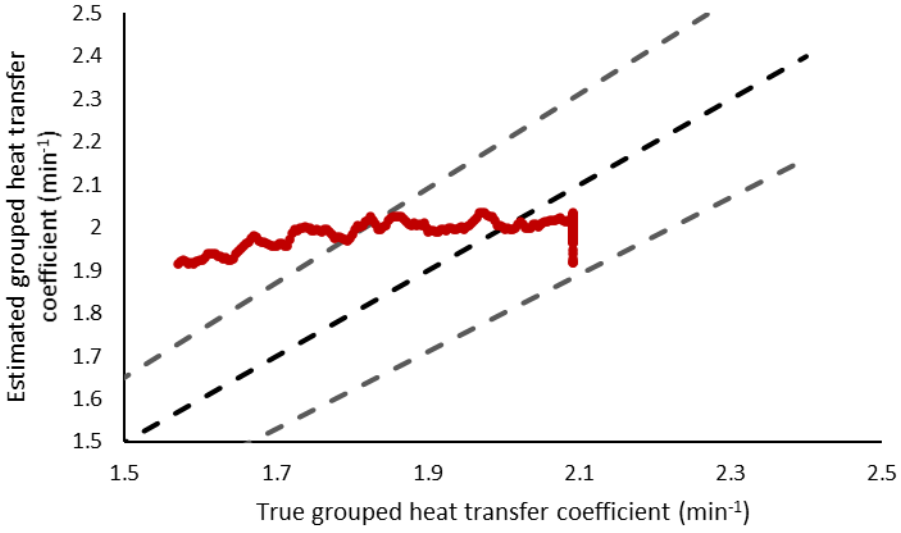

Figure 46: Parity plot for grouped heat transfer coefficient using dPLS where heat transfer coefficient error is active with 10% relative error margins

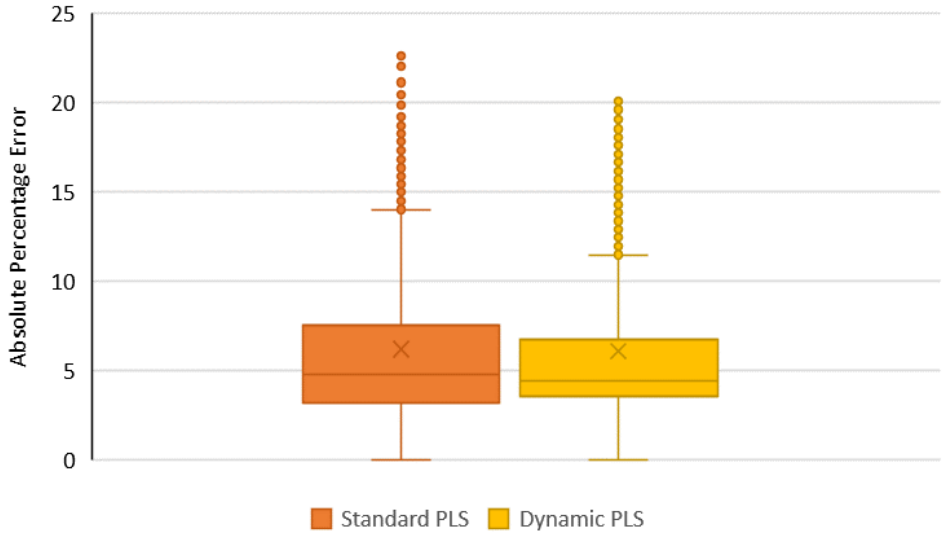

Figure 47: Box and whisker plot comparing APE for standard PLS and dynamic PLS where overall heat transfer coefficient fault is active

The error of the PLS models in estimating heat transfer coefficient for the test case is quantified through the MAPE measure. The MAPE was calculated as 5.95% and 5.74% for the PLS and dPLS models respectively. A box and whisker plot was generated to compare the variance of the APE values used in the MAPE and is shown in Figure *47* above.

When the plot is studied it is seen that the dPLS shows less variance in the predictions than the PLS model. It is seen that there are outliers experienced in the PLS model where the magnitude of the error is larger than those experienced by the dPLS model. This indicates poor predictions by the PLS model for those observations. These errors can be attributed to the nature of the observation as it could be that the model performs poorly on these observations due to the process variable values for those observations. The dPLS provides more accurate predictions, as by considering a range of observations which precede the observations the impact of one abnormal observation is not as great.

The model parameters are used to predict the output variable using the model ordinary differential equations. The predicted outputs are compared to the true values for the outputs obtained from the simulation of the jacketed CSTR. The temperature predictions are compared in Figure 48 and the concentration predictions are compared in Figure 49.

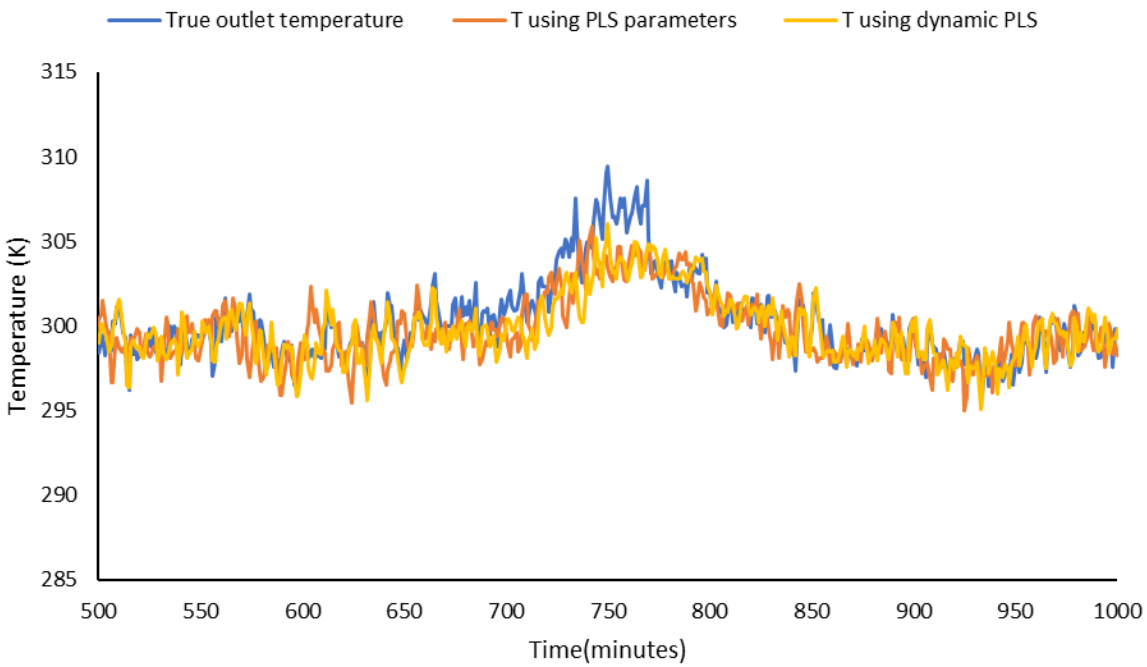

Figure 48: Outlet temperature predictions over span of 500-1000 minutes with heat transfer coefficient fault active

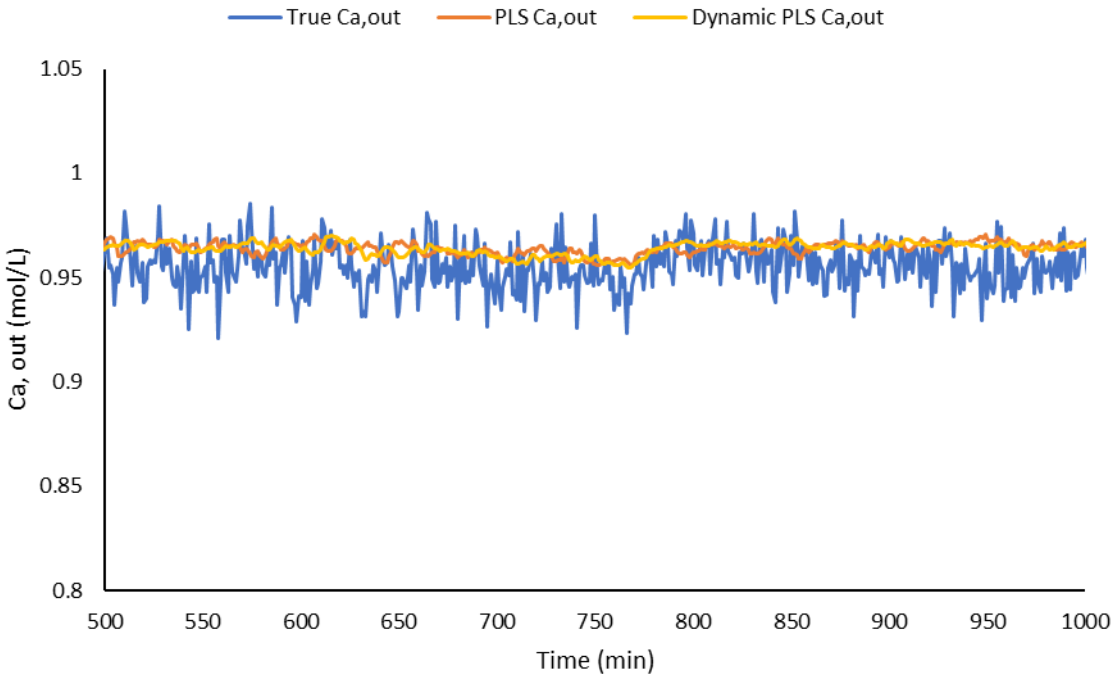

Figure 49: Outlet concentration prediction over a time window of 500-1000 minutes with heat transfer coefficient fault active

As seen above, the PLS and dPLS model predictions for the outlet temperature do display the changes in the outlet temperature however the model predictions do not capture the magnitude of the change. The outlet concentration estimates do not show as much variance as the true values however it does appear to reflect the changes in the concentration values. This indicates that the hybrid models using the standard and dynamic PLS provide reasonable output estimates for the heat transfer coefficient fault case. The error for both models for the output predictions are quantified using the MAPE values and are included in Table 9. The small MAPE values indicates that the PLS model predictions are accurate which is evident in the comparison plots.

Table 9: MAPE for outlet predictions where heat transfer coefficient fault is active

Outlet variable	Standard pls	Dynamic pls
Temperature	0.839	0.899
Concentration	0.916	1.16

Based on the hybrid model performance comparisons both the PLS and dPLS models perform similarly. It appears that when the PLS and dPLS parameter estimates are used to predict the outlet conditions, the predictions provide reasonable estimates of the true outlet values.

In the r-SVM and r-kSVM models it is seen that by introducing the model parameters to the classification models the fault identification performance increases. As both PLS models are proposed as alternate methods of obtaining the process parameters ideally the performance of the fault identifiers would be comparable to that of the r-SVM and r-kSVM models. The performance of the PLS and dPLS based fault identification models applied to the catalyst deactivation-inlet concentration fault pair are compared in Figure 50.

The sensitivities of the PLS-SVM and dPLS-SVM models are 0.834±0.017 and 0.904±0.026 respectively. When the standard SVM performance is considered the dPLS-SVM model displays a higher sensitivity than the PLS-SVM model, one expects a slightly higher sensitivity when using the dynamic PLS model as the model demonstrated a higher prediction accuracy.

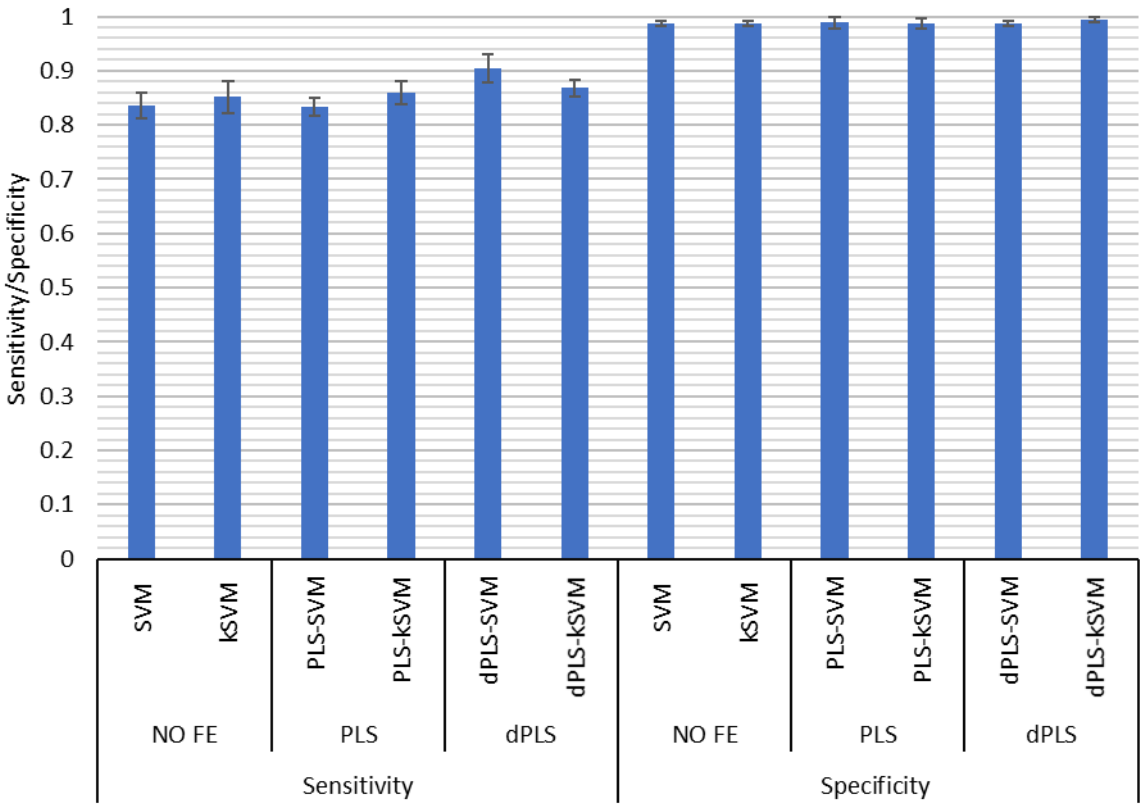

Figure 50: Sensitivity and specificities for PLS and dPLS models for catalyst deactivation-inlet concentration fault pair with catalyst deactivation fault active

The increase in sensitivity experienced is higher than one would expect, however the improvement in prediction accuracy was not as great as the increase in sensitivity. This shows that a slight increase in prediction accuracy can lead to a larger increase in sensitivity. The increase in sensitivity is due to the additional time lagged variables resulting in more accurate parameter estimates which provides a clear separation of the faulty data from the normal operation data. The more accurate parameter estimates alter the support vector locations which changes the position of the separating hyperplane to provide a clear separation between fault cases. The performance of this fault identifier configuration is comparable to that of the r-SVM model which represents the theoretical ideal performance for the considered fault pair.

The sensitivities of the PLS-kSVM and dPLS-kSVM models are 0.860±0.022 and 0.869±0.015 respectively. When the sensitivities of the kSVM fault identification methods are compared it is seen that the when the PLS-kSVM model is used there is a slight increase in the sensitivity. When the dPLS-kSVM model is used an increase in sensitivity is seen with the sensitivity being slightly larger than the PLS-kSVM model. This is likely due to the model parameters estimated

by the PLS models fed to the classifier resulting in a clear separation between the fault case and the normal operation case.

The specificities of the PLS-SVM and dPLS-SVM models are 0.988±0.004 and 0.987±0.004 while the PLS-kSVM and dPLS-kSVM specificities are 0.990±0.005 and 0.990±0.005. When the specificities of the SVM and kSVM models are considered, that there are only slight differences in the magnitudes of the fault identification models. When compared to the r-SVM and r-kSVM models the PLS fault identifiers show slightly lower specificities.

The ability of the fault detection method to identify the catalyst deactivation fault at different fault magnitudes is evaluated using sensitivity curves as shown in Figure 51. Both the kSVM and PLS-kSVM model displays significant scatter in the sensitivity measures at different fault magnitudes. This indicates that the performance of the model is dependent on the dataset considered and does not directly correlate to the magnitude of the considered error. This could be due to the dataset available determining the position of the support vectors, with some datasets resulting in different support vectors selected which effect the performance of the classifier and in turn the sensitivity. When the dPLS model performance is evaluated, it is seen that as the fault magnitude increases the dPLS sensitivity steadily increases. It is also seen that for the majority of the fault magnitudes considered the dynamic PLS models display a higher sensitivity. This is the performance that one would expect.

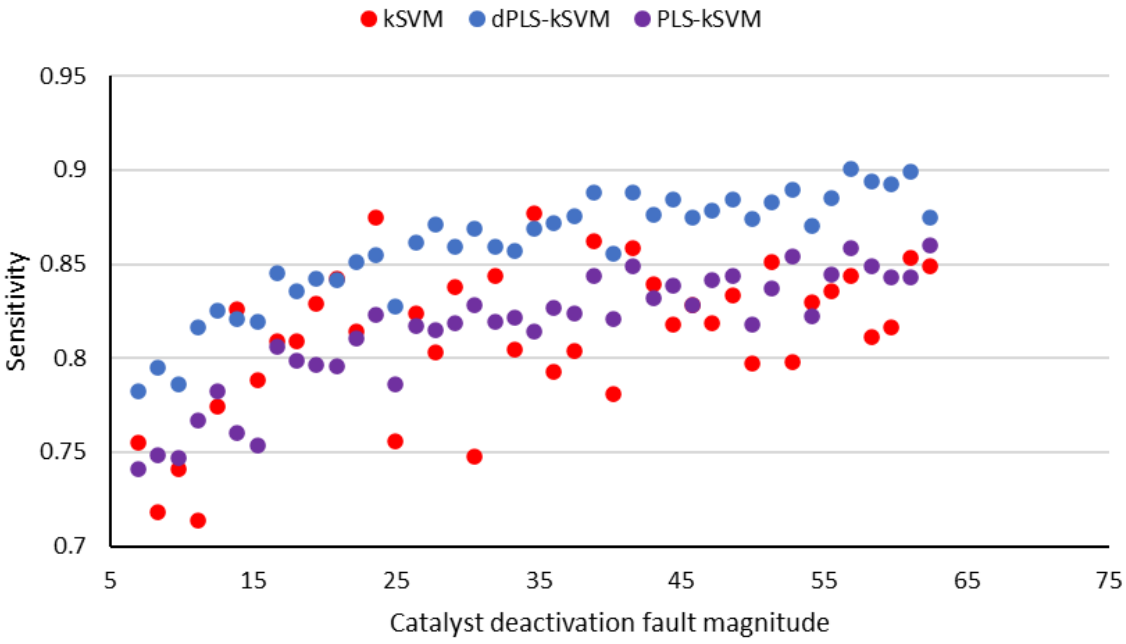

Figure 51: Sensitivity curves for catalyst deactivation fault at different magnitudes

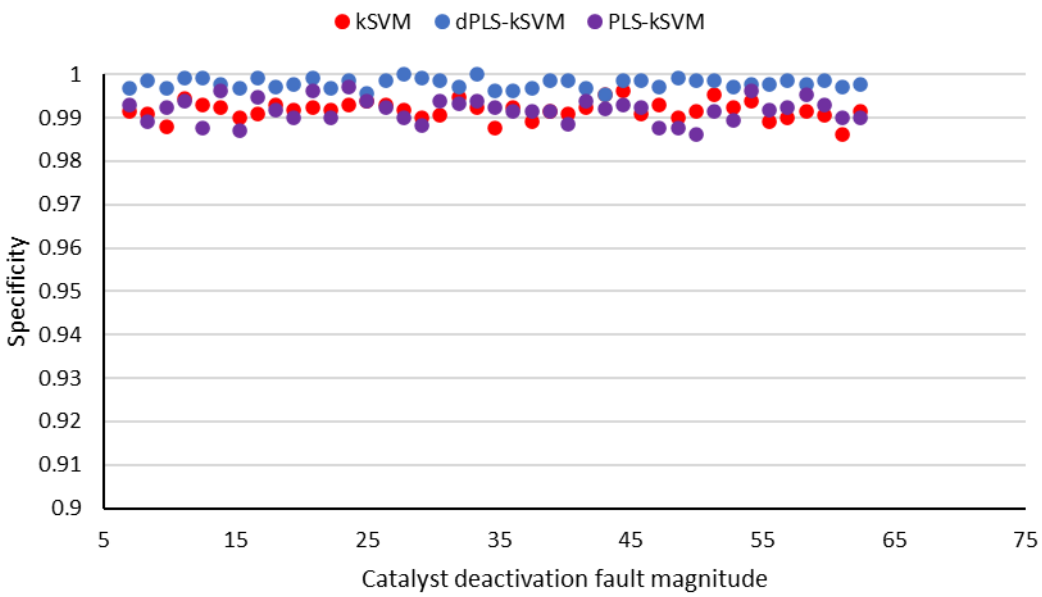

Figure 52: Specificity curves for catalyst deactivation faults at different magnitudes

The specificity values at several catalyst deactivation fault magnitudes are shown in the specificity curve illustrated by Figure 52 above. As seen the specificities do not show great variations, only showing slight noise in the readings as the fault magnitude increases. It also seen that the dPLS fault identification specificities are only slightly larger than the specificities than the other fault identification methods considered. This could be related to the increase in sensitivity experienced when the dynamic PLS fault detection method is used. By introducing the hybrid modelling components an increase in sensitivity is noted and this could possibly lead to a decrease in specificity as it could lead to increase in false fault detections. However, as the specificity does not show large variations the introduction of the PLS models do not increase false fault detection rates, even as the fault magnitudes increase.

The ability of the fault identification method to discern between two process faults is evaluated by training the fault identification model on normal operation, catalyst deactivation fault and heat transfer fault data. As before, the performance of the fault identification model is quantified and compared using the sensitivity and specificity measures for the PLS- and dPLS-SVM and kSVM models as seen in Figure 53.

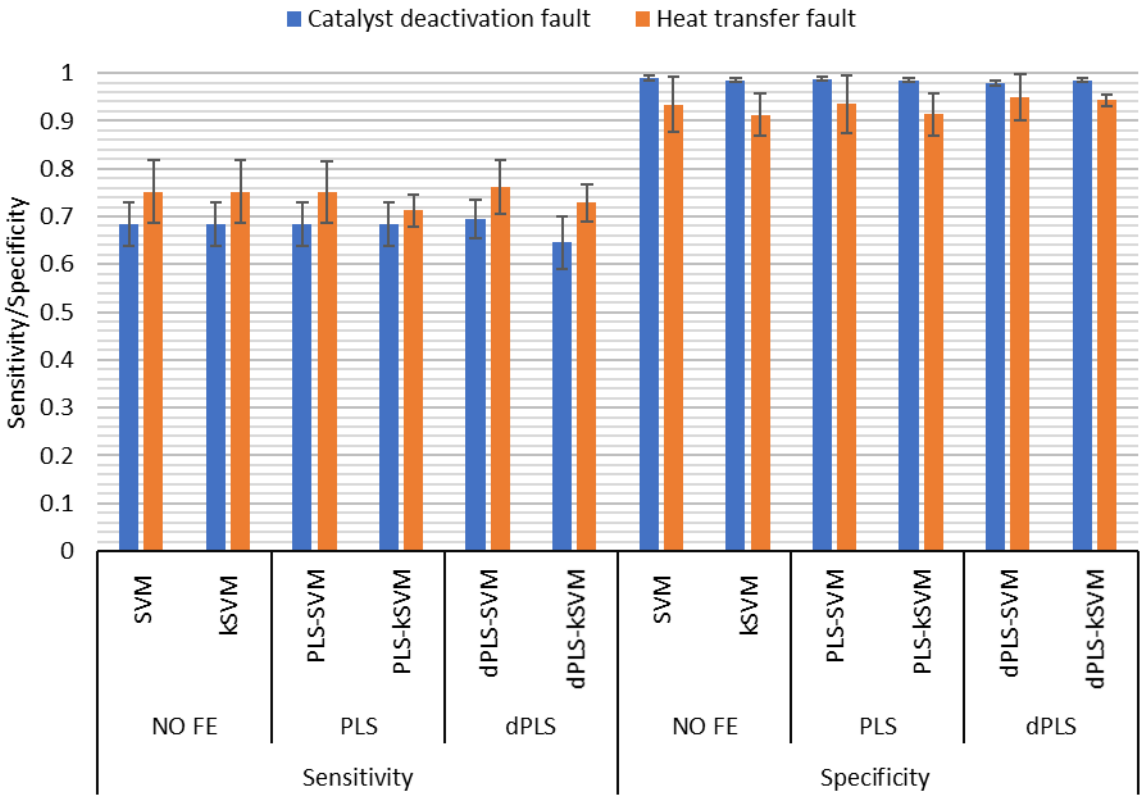

Figure 53: Sensitivity and specificity for PLS and dPLS models for heat transfer – catalyst deactivation fault pair

The sensitivities of the PLS-SVM and dPLS-SVM models when the catalyst deactivation fault is active are 0.684±0.044 and 0.695±0.041 respectively. The PLS-kSVM and dPLS-kSVM sensitivities are 0.683±0.045 and 0.645±0.054. When the fault identification models are applied to the test datasets in which the catalyst deactivation fault is active it is seen that most of the PLS and dPLS models do not improve the performance of the fault identification models. Only the dPLS-SVM model shows an increase in performance, it is even noticed that when the hybrid models are applied to the kSVM classifiers the sensitivities decrease. This is not what one would expect, as when the models were applied to the catalyst deactivation-inlet concentration fault pair an increase in sensitivity was seen.

The decrease in sensitivities indicates that feeding the additional information not only does not add any value to the identification of the catalyst deactivation fault, but it also actually decreases the performance of the model in the kSVM models. The additional parameters cause the different fault cases to appear more similar thus decreasing the classification efficiency.

The specificities of the PLS-SVM and dPLS-SVM models are 0.988±0.004 and 0.979±0.005 while the PLS-kSVM and dPLS-kSVM specificities are 0.986±0.005 and 0.985±0.005. The specificity measures do not improve based on which fault identification method is used which is similar to the performance shown in the previous testcase.

The fault identification methods were also applied to test datasets where the overall heat transfer coefficient fault was active, and the sensitivities and specificities are also shown in Figure 53. The sensitivities of the PLS-SVM and dPLS-SVM models when the catalyst deactivation fault is active are 0.751±0.065 and 0.761±0.056. The PLS-kSVM and dPLS-kSVM sensitivities are 0.712±0.033 and 0.728±0.039. The PLS and dPLS fault identifiers performances are lower than that of the r-SVM and r-kSVM models. The classification models which do not use kernel transformation have shown better performance than the classification models which do. This is evident by the models which use the SVM classifier having both higher sensitivities and specificities.

One would expect the hybrid models not to improve on the sensitivities of the classification models as the PLS models do not track the changes in overall heat transfer coefficient as seen in Figure 43. The dPLS model display a slight increase in performance when compared to the SVM and kSVM fault identifiers. Although the effects of the heat transfer fault are accounted for by the reaction rate as seen in Figure 44 the reaction rate estimates, although different from those of true values for the fault case, resembles the values at normal operation. As the estimates does not result in the heat transfer fault case being more clearly separable from the other cases the locations of the support vectors in the classifier do not change. This results in no increase in the sensitivity of the fault identification model being experienced.

The specificities of the PLS-SVM and dPLS-SVM models are 0.935±0.060 and 0.949±0.049 while the PLS-kSVM and dPLS-kSVM specificities are 0.913±0.045and 0.943±0.012. The specificity values when the heat transfer fault is active do change between the fault identification models. This is a similar performance to that experienced by the r-SVM and r-kSVM (shown in Figure 34). The specificity values however are not significantly different from those shown by the SVM and kSVM models. The dPLS-SVM model however shows a higher specificity than the other models when the heat transfer fault is active. This increase in specificity is likely due the parameters fed to the classifier providing clear bounds for the normal operation class. The parameter estimates predicted by the dPLS model shows less

scatter (due to considering observations over a time window) than the PLS model resulting in clearer boundaries for the normal operation class. This in turn decreases the false fault identification rate which increases specificity. The overall increase in performance is not as great as that seen when using the non-linearly regressed parameter estimates as the sensitivity does not increase.

The models which made use of SVM as the classification model showed better performance identifying the heat transfer fault than the kSVM models. Thus, the ability of the SVM fault identification models to identify the heat transfer fault at different fault magnitude is evaluated using a sensitivity curve seen in Figure 54. The dPLS-SVM shows the highest overall performance as for the majority of fault magnitudes the sensitivity is higher than that of the other fault identification models which agrees with the results discussed above. It is seen that the SVM and PLS-SVM show similar sensitivities for all magnitudes considered which is due to the PLS model not tracking the changes in heat transfer fault. There is scatter noticeable between the sensitivity measurements, this is likely the different data entries selected as support vectors at each fault magnitude. The training data used in each cross-validation case will have an impact on the performance of the classifier.

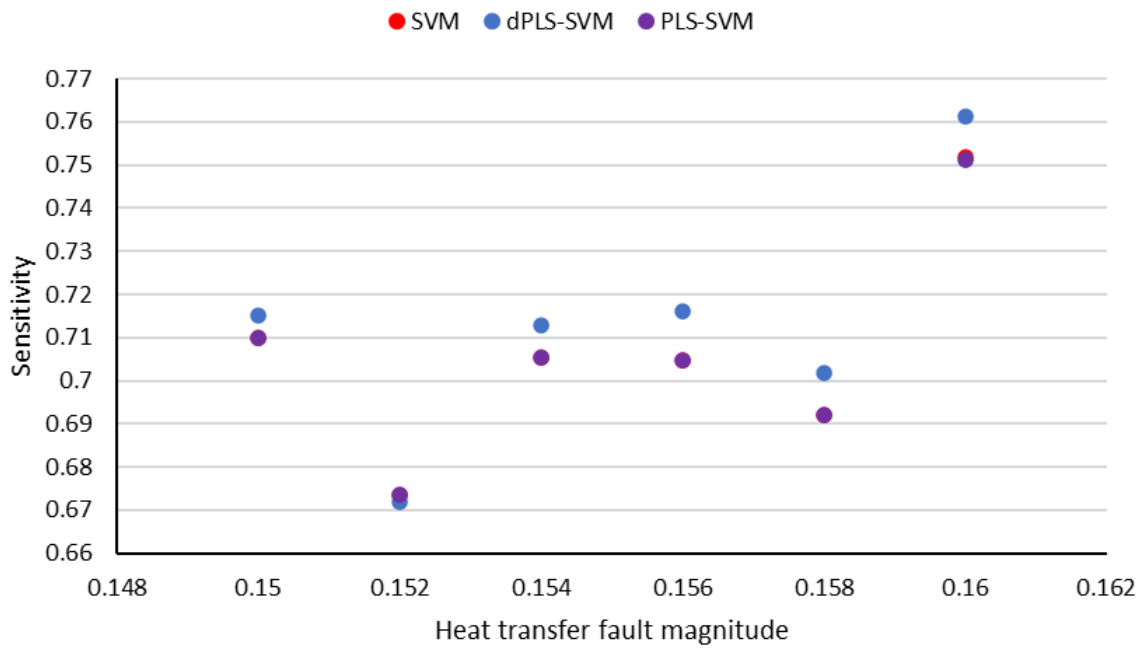

Figure 54: SVM models sensitivity curves for the heat transfer fault at different fault magnitudes

The specificity curve for the heat transfer fault is shown in Figure 55 below. As seen the dPLS-SVM model specificities shows higher performance than the other SVM fault identifiers. This agrees with the findings discussed above. Based on the results presented above the best performing shortcutting hybrid model is the dPLS-SVM model.

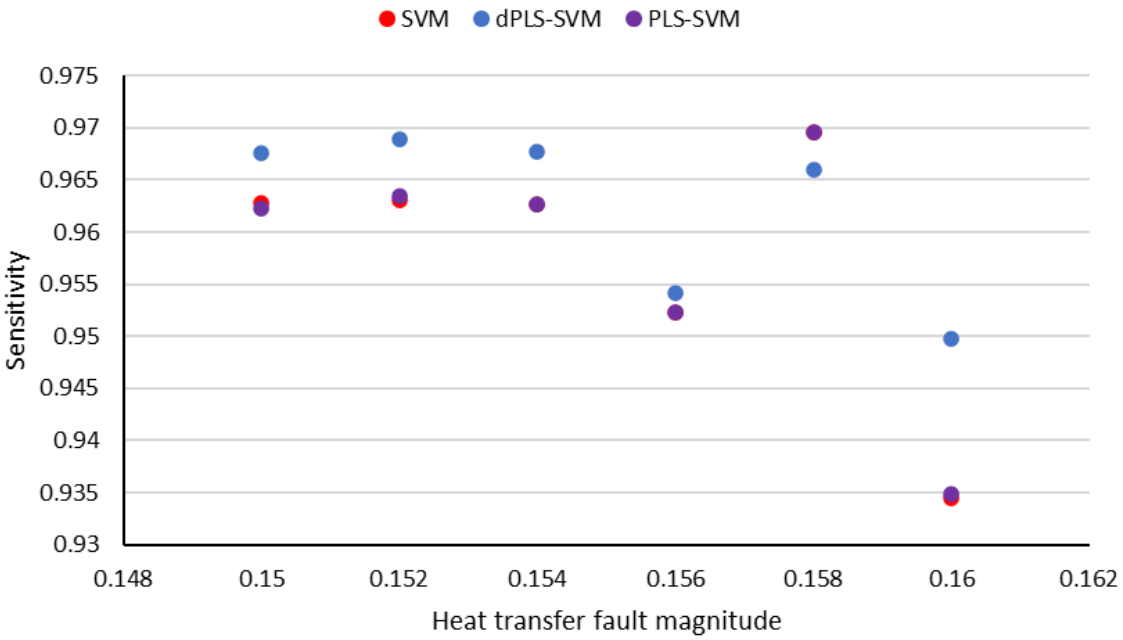

Figure 55: SVM models specificity curves for the heat transfer fault at different fault magnitudes

5.5 Recursive PLS

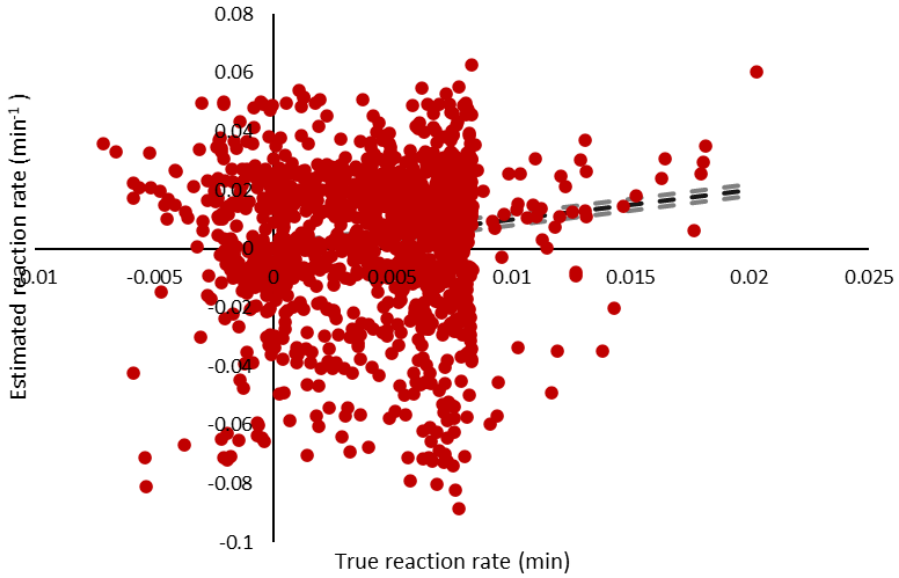

Figure 56: Parity plot using recursive PLS with 10% relative error bars

A parity plot is constructed using the recursive PLS model as shown in Figure 56 below for the catalyst deactivation-inlet concentration fault pair. It is clearly visible that there is significant scatter surrounding the 45° line. This indicates that for the catalyst deactivation case considered the recursive PLS model does not predict the test set well at all. It is seen that by having many data entries lying below the x-axis the model tends to overestimate not only the magnitude of the decrease in the reaction rate due to the catalyst deactivation fault but also the portion of data which has the fault active. Based on the poor predictive performance of the r-PLS model one would not expect the parameter estimates will improve the fault identification.

To evaluate the fault identification performance using the rPLS model is evaluated using the rPLS-kSVM model for the catalyst deactivation-inlet concentration fault pair. The sensitivity and specificity values are shown in Figure 57 below. As seen, there is a noticeable decrease in both the sensitivity and specificity values which can be attributed to the poor prediction capabilities of the rPLS model. Due to the poor performance of the rPLS fault identifier it is not considered further in the study.

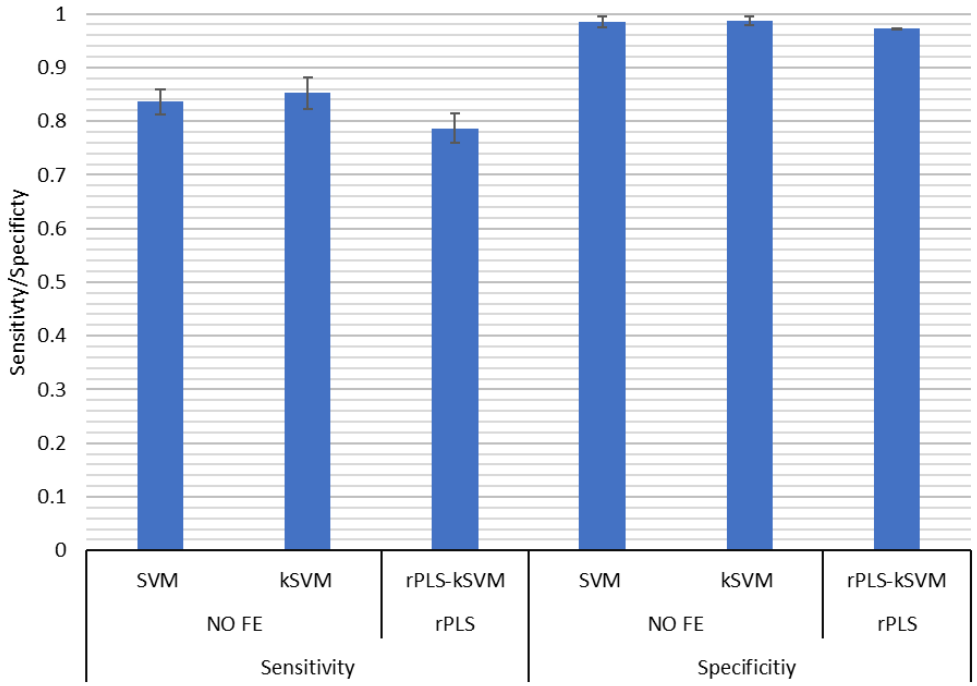

Figure 57: rPLS-kSVM model for catalyst deactivation-inlet concentration fault with catalyst deactivation fault active

5.6 Overall comparison

The performances of the fault identification models can be compared to determine which model configuration provides the best performance. The performances of the best performing models based on the comparisons performed in the preceding sections of this chapter are compared. The best performing models were found to be the r-SVM and dPLS-SVM models and are compared to the SVM model which acts as the baseline comparison model as well as the kSVM model which showed better performance. The performances of the models for the catalyst deactivation-inlet concentration fault pair are shown in Figure 58. The figure displays an enlarged section of the sensitivity and specificity comparison plot to better compare the performances of the models.

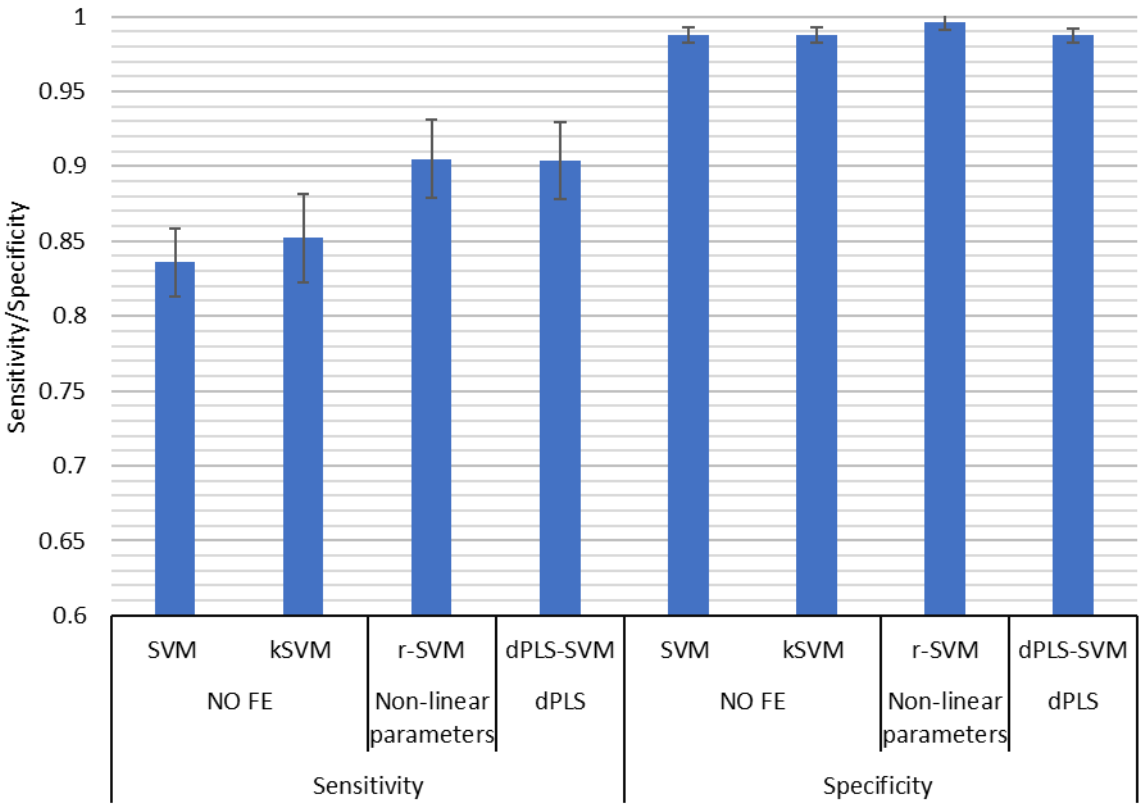

Figure 58: Overall comparison for catalyst deactivation-inlet concentration fault pair for the r-SVM and dPLS-SVM models with catalyst deactivation fault active

From the comparison both the r-SVM and the dPLS-SVM models perform similarly for the considered fault pair. Both models would be preferable to the standard SVM fault identification model for this fault pair with the increase in performance due to the additional parameters fed to the classifier resulting in a clearer separation of the fault cases. For the fault pair considered above the dPLS-SVM would be the preferable fault identifier due to the shorter execution time when applied online. Real time non-linear regression is more time consuming than using a PLS model to estimate model parameters based on incoming data. Thus, implementing non-linear regression in an online fault identification technique would result in delays in the fault identification if the time interval between measurements is shorter than the execution time of the regression.

The performance of the models for the catalyst deactivation-heat transfer fault pair are compared in Figure 59 below. When the catalyst deactivation fault is identified it is seen that the models perform similarly with the sensitivities not showing much variation between the models. This indicates that the different model configurations do not improve the ability of the

fault identifier to correctly identify the catalyst deactivation fault. The differences in performances are shown in the specificity measures with the hybrid models (r-SVM and dPLS-SVM) showing lower specificities than the classifier only models. This indicates that introducing the additional parameters makes the models more likely to incorrectly identify normal operation instance as being faulty. The dPLS-SVM model shows higher specificity than the r-SVM model thus showing better performance than the r-SVM when identifying the catalyst deactivation fault.

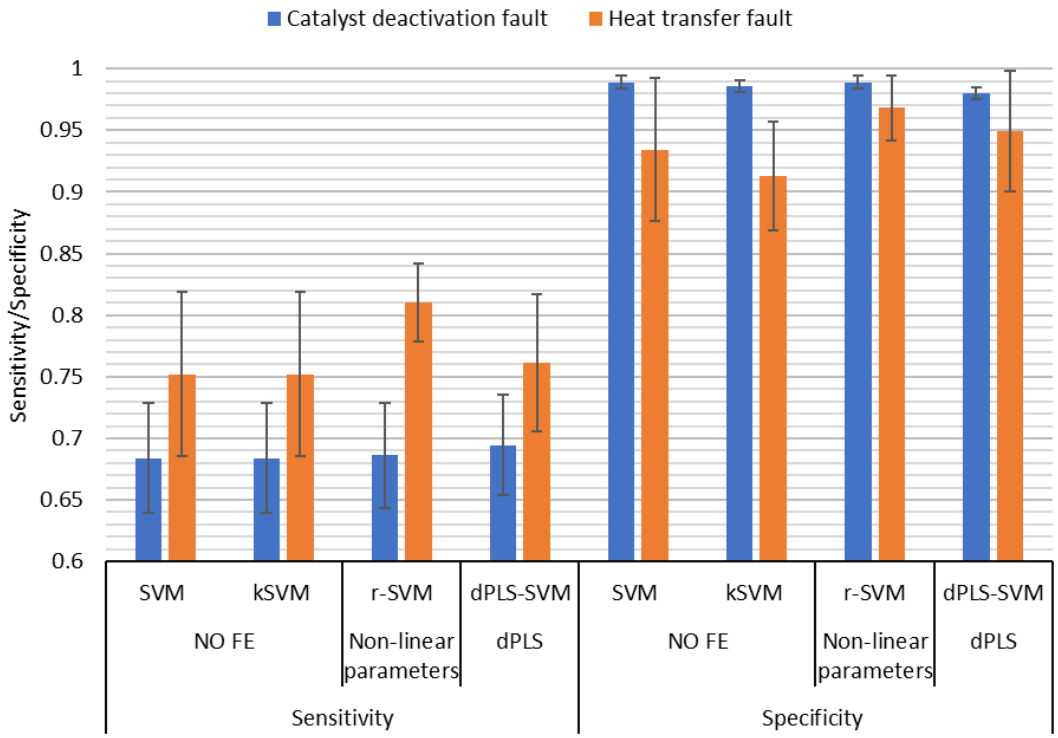

Figure 59: Overall comparison for catalyst deactivation-heat transfer fault pair for the r-SVM and dPLS-SVM models

When identifying the heat transfer fault, the r-SVM model displays a higher sensitivity than the other models considered with a 5% difference between the sensitivities. This due to the parameter estimates included in the r-SVM tracking the change in heat transfer coefficient which is absent in the SVM only models and the dPLS model does not accurately track the changes in the parameter. When the specificities are considered the r-SVM model shows a lower overall value than the dPLS-SVM model which is due to the model being more likely to identify the fault (indicated by the higher sensitivity) thus misclassifying normal operation observations as faulty. The difference however is not large (approximately 2% deviation) which would

correspond to 8 normal operation misclassifications for a test set of 400 observations. Based on the specificities the false alarm rate for the r-SVM (which is indicated by 1-specificity) model is 3% and the dPLS-SVM is 5%. This translates to the r-SVM having a false alarm rate which is 20% lower than the dPLS-SVM model. Based on the higher heat transfer fault identification performance and comparable performance to rest of the models for the other performance measures the r-SVM model is identified as the best performing model.

To better visualise and evaluate the performance of the fault identifier a confusion matrix is generated. The confusion matrix for the r-SVM model applied to the catalyst deactivation-heat transfer fault pair is shown in Figure 60. It is seen that the model correctly identifies a large portion of both faults correctly which explains the high sensitivities shown. An important point to note is that the identifier does not incorrectly identify one fault as the other (i.e identifying heat transfer fault as catalyst deactivation fault). This an important aspect of the fault identifier as it ensures that the correct rectification measures are applied to rectify the appropriate fault. If the wrong corrective measures are applied to a system, it could result in the process deviating further from desired specifications and performance. This also indicates that any misclassification of faults (which results in lower sensitivities) are due to the fault detection aspect of the model (i.e., distinguishing between normal and fault conditions) and not due to incorrect distinguishing between faults. The multiclass SVM approach results in 100% separation efficiency between the two process faults. Thus a possible method of improving the fault identification aspect of the model is by introducing a separate fault detection model before fault identification is applied as was done in the work by (Heo & Lee, 2018) for example.

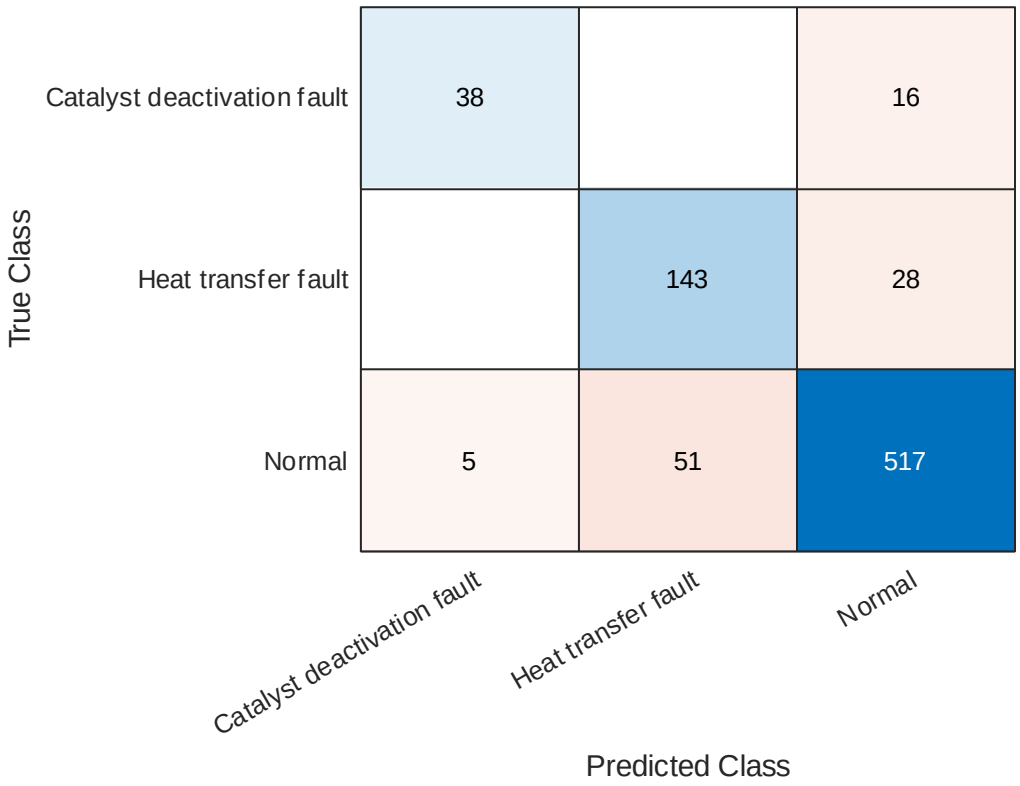

Figure 60: r-SVM confusion matrix for catalyst deactivation-heat transfer fault pair

Based on the results discussed the hybrid modelling feature extraction is a viable method of improving the performance fault identification models as the r-SVM model structures shows an improvement over the SVM classifiers. This is clearly seen as the introduction of the estimated parameters increases the performance of the fault identification models. As the PLS models do not track the changes in the overall heat transfer coefficient well another databased model might lead to better performance. A possible data-based model which can be used is the ANN as it has performed well in hybrid models such as the work done by (Bhutani *et al.*, 2006)

6 CONCLUSIONS

Fault detection and identification models are critical in maintaining plant operating and product quality. Many fault identification techniques have been developed to address these concerns however many fault identifiers operate by identifying the appropriate fault once a fault has been detected. A multiclass SVM classifier is proposed as the fault identification method and the simultaneous fault detection-identification is achieved by including normal operation data in the training dataset.

A point of consideration is that additional process information can be obtained which could result in clearer separation and clustering of fault laden observations. Thus, feature extraction and selection methods have been applied to fault detection models which showed an increase in performance. A hybrid modelling feature extraction method in which model parameters are estimated and used in the fault identification was proposed. The PLS, dPLS and rPLS models are considered as possible data-based models which can be used in the hybrid models.

A non-isothermal jacketed CSTR model was developed and used as test case model on which the fault identification models can be applied, and their performance recorded. Process faults were developed and applied to the test case model with two faults applied to parameters which are estimated, and one fault applied to a process variable which is neither recorded nor measured. The hybrid model was implemented to estimate model parameters based on incoming process data. The data-based models were trained using parameter values obtained using non-linear regression. The fault identification models were trained on process fault pairs and tested on test cases to evaluate the performance of the fault identification models. The sensitivity and specificity measures were used to quantify the performance of each fault identification model on each process fault.

As the SVM classifier can employ kernel transformations it needs to be established if the kernel transformation will improve the performance of the fault identification classifiers. A preliminary evaluation of the models was done it was found the kernel improved the performance of the SVM. Thus, kernel SVM classifiers were considered in the study. The SVM and kSVM models were applied to the catalyst deactivation-inlet concentration fault and the catalyst deactivation-heat transfer fault. It was found the kSVM showed slightly better performance. It was also seen that based on these results the multiclass SVM approach could be used for fault identification.

Using extracted features in the fault identification is proposed as method of improving the performance of the fault identification models. The performance of the fault identification models using the non-linearly regressed parameter estimates were evaluated to test whether the extracted features improve the fault identification performance. It was found that both the r-SVM and r-kSVM models showed an increase in performance with the increase due to the non-linearly regressed parameters being able to track the decrease in parameter values due to the process faults. The r-SVM model showed better performance than the r-kSVM model with the increase in performance due to the observations selected as support vectors with the r-SVM selected support vectors providing better separation between faults. This shows that utilising extracted features in the form model parameters does indeed improve the performance of the fault identification models

The PLS, dPLS and rPLS models were proposed as a method of estimating model parameters based on incoming data. Parity plots were generated using the PLS and dPLS models, both models showed comparable performance in estimating the model parameter when the catalyst deactivation fault is active. The dPLS model displayed slightly higher accuracy for the considered test cases. The performance of models in estimating the parameters when the heat transfer fault is active was evaluated. It was seen that the heat transfer coefficient estimates did not reflect the changes due to the fault. The effect of the was accounted for in the reaction rate estimates. Thus, the PLS and dPLS models do not provide the most accurate model parameter estimates.

The ability of the fault identifiers to identify the catalyst deactivation fault in the catalyst fault – inlet concentration fault pair was tested. When the PLS model was implemented as the model parameter estimator no significant increase performance seen. The dPLS fault identification models showed an improvement in performance when compared to the SVM and kSVM models. The dPLS-SVM model showed a larger increase in sensitivity.

The specificity of the PLS-SVM showed a decrease when compared to the other models which could be attributed to the higher fault detection rate. This results in a higher false normal identification rate which results in a decrease in specificity. The specificities for the other models were determined, and it was found that the values do not vary significantly between the models. This indicates that when the rest of the models were compared that none of the models were more susceptible to false normal identification than the other. It was found that under

these conditions the dPLS model is a sufficient substitute for the non-linear regression as the dPLS fault identification models have performance comparable to the r-SVM and r-kSVM models.

The fault identifiers were then trained on the catalyst deactivation – overall heat transfer coefficient fault pair to evaluate the capability of the fault identifiers to discern between two different process faults. When evaluated on the catalyst deactivation fault the PLS fault identification models showed no improvement over the SVM and kSVM. The dPLS-SVM model shows a significant increase in performance.

The fault identification models were applied to the overall heat transfer coefficient fault test sets and the sensitivities and specificities were obtained. It was seen that the PLS and dPLS did not improve the performance of the fault identification models under these conditions. It is seen that the models are not only able to detect the fault but are also able to differentiate between the heat transfer coefficient and catalyst deactivation faults.

The specificities were determined, and it was found that all the specificities were similar with the dPLS model showing slight increase. This indicates that for all faults considered the fault identification models are not susceptible to false fault identifications. It was seen that the SVM models showed better performance overall than the kSVM models with the dPLS-SVM model showing the best performance. Based on the results the dPLS model could be used an alternate method of estimating parameter estimates and replacing non-linear regression. The dPLS-SVM model could be used as a fault identification model which improves on the performance of the SVM and kSVM fault identification models.

The rPLS fault identification model was then considered. When a parity plot was generated using the recursive PLS model it was found that the PLS model does not perform well on the considered datasets. It was thus expected that the rPLS fault identification models would not perform well. The performance rPLS-kSVM model was evaluated and was shown to have poor performance, performance worse than the SVM and kSVM fault identification models. Thus, the rPLS fault identification models were not considered further and would not be an appropriate model to use as a parameter estimation model.

The r-SVM and dPLS-SVM models were compared to determine the best performing fault identification model. It was seen that the r-SVM showed higher sensitivity and specificity than the dPLS-SVM model. Therefore, the r-SVM was the overall best performing model. It was

seen that hybrid modelling feature extraction is a feasible method of improving fault identifier performance. As the PLS models did not capture the change in parameters values as well as the non-linear regression another data-based model can be used which better tracks the changes in parameter values.

6.1 Recommendations based on current work

Based on the conclusions of the study as discussed above the following recommendations are proposed to possible improve on the work performed:

1. SVM and kSVM classifiers should be considered when designing and implementing fault identification methods
2. For the CSTR process considered in this study the SVM classifier should be implemented
3. If a process model is available for a system, the model-based feature extraction method should be considered as method of improving fault identification
4. For the parameter estimation the dPLS model should be used instead of the PLS model
5. If possible real time non-linear regression should be used as parameter estimation as the regression provides the most accurate parameter estimates
6. For the considered process the r-SVM fault identification model should be used and can be used as a first attempt at improving fault identification model performance for other processes

6.2 Recommendations for future work

Based on the results and conclusions of the study the following recommendations for future work which could for the basis of a new study are proposed:

1. Applying the hybrid modelling based feature extraction fault identification model to more realistic process datasets and models such as the Tennessee-Eastman process to evaluate the applicability of the model to real world processes

2. Evaluate the performance of the fault identification model where a different data-based model is used in hybrid modelling such as the ANN model as the ANN model has been proven to perform well in hybrid models

3. Consider the use of relevance vector machine classifiers in lieu of the support vector machine classifiers used. The relevance vector machine classifier makes use of probabilities thresholds for classification. The probabilities can then be used to further evaluate the performance of the classifier

7 REFERENCES

Addo, P. 2019. Adaptive Process Monitoring using Principal Component Analysis and Gaussian Mixture Models.

Aguiar, H.C. & Filho, R.M. 2001. Neural network and hybrid model: A discussion about different modeling techniques to predict pulping degree with industrial data. *Chemical Engineering Science*. 56(2):565–570.

Aizerman, M.A., Braverman, E.A. & Rozonoer, L. 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control,*. 25(6):821–837.

Angeli, C. 2010. Diagnostic Expert Systems: From Expert's Knowledge to Real-Time Systems. In *Advanced Knowledge Based Systems: Model, Applications and Research*. 50–73.

Bellman, R.E. 1961. *Adaptive control processes*. 1st ed. Princeton,NJ: Princeton University Press.

Bhutani, N., Rangaiah, G.P. & Ray, A.K. 2006. First-principles, data-based, and hybrid modeling and optimization of an industrial hydrocracking unit. *Industrial and Engineering Chemistry Research*. 45(23):7807–7816.

Bishop, C.M. 2006. *Pattern Recognition and Machine Learning*. 1st ed. New York, N.Y.: Springer.

Boser, B.E., Guyon, I.M. & Vapnik, V.N. 1992. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. 144–152.

Che Mid, E. & Dua, V. 2017. Model-Based Parameter Estimation for Fault Detection Using Multiparametric Programming. *Industrial and Engineering Chemistry Research*. 56(28):8000–8015.

Christanni, N. & Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 1st ed. Cambride: Cambridge University Press.

Conlin, J., Peel, C. & Montague, G.A. 1997. Modelling pressure drop in water treatment. *Artificial Intelligence in Engineering*. 11(4):393–400.

Estrada-Flores, S., Merts, I., De Ketelaere, B. & Lammertyn, J. 2006. Development and validation of "grey-box" models for refrigeration applications: A review of key concepts. *International Journal of Refrigeration*. 29(6):931–946.

Fogler, H.S. 2006. *Elements of chemical reaction engineering*. 4th ed. (Prentice Hall PTR international series in the physical and chemical engineering sciences). Upper Saddle River, NJ: Prentice Hall PTR.

Geladi, P. & Kowalski, B.R. 1986. Partial least-squares regression: a tutorial. *Analytica Chimica Acta*. 185:1–17.

Gertler, J. 1997. Fault detection and isolation using parity relations. *Control Engineering Practice*. 5(5):653–661.

Hastie, T., Tibshirani, R. & Friedman, J. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Predicition*. 1st ed. New York: Springer.

He, Q.P. & Wang, J. 2007. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*. 20(4):345–354.

Heo, S. & Lee, J.H. 2018. Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine*. 51(18):470–475.

Hotelling, H. 1931. The generalization of Student's ratio. *Annals of Mathematical Statistics*. 2(3):360–378.

Hu, G., Mao, Z., He, D. & Yang, F. 2011. Hybrid modeling for the prediction of leaching rate in leaching process based on negative correlation learning bagging ensemble algorithm. *Computers and Chemical Engineering*. 35:2611–2617.

Isermann, R. 2005. Model-based fault-detection and diagnosis - Status and applications. *Annual Reviews in Control*. 29(1):71–85.

Isermann, R. 2006. *Fault-diagnosis systems: an introduction from fault detection to fault*

tolerance. 1st ed. Berlin: Springer.

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2017. *An Introduction to Statistics Learning with Application in R*. 8th ed. New York: Springer.

Jia, R. da, Mao, Z. zhong, Chang, Y. qing & Zhao, L. ping. 2011. Soft-sensor for copper extraction process in cobalt hydrometallurgy based on adaptive hybrid model. *Chemical Engineering Research and Design*. 89(6):722–728.

Kohavi, R. 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference of Artificial Intelligence*. 1137–1145.

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

Lee, D.S., Jeon, C.O., Park, J.M. & Chang, K.S. 2002. Hybrid neural network modeling of a full-scale industrial wastewater treatment process. *Biotechnology and Bioengineering*. 78(6):670–682.

Van Der Maaten, L.J.P., Postma, E.O. & Van Den Herik, H.J. 2009. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*. 10(1):1–41.

Miljković, D. 2011. Fault Detection Methods : A Literature Survey. In *Proceedings of the 34th International Convention MIPRO*. 750–755.

Onel, M., Kieslich, C.A. & Pistikopoulos, E.N. 2019. A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the Tennessee Eastman process. *AIChE Journal*. 65(3):992–1005.

Platt, J. 2000. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Vol. 10. A. Somala, P. Bartlet, B. Schölkopf, & D. Schuurmans (eds.). Cambridge: MIT Press *Advances in Large Margin Classifiers*. 61–74.

Qin, S.J. 1998. Recursive PLS algorithms for adaptive data modeling. *Computers chem. Engng*. 22(5):503–514.

Ricker, N.L. 1988. The use of biased least-squares estimators for parameters in discrete-time

pulse-response models. *Industrial and Engineering Chemistry Research*. 27(2):343–350.

Rosipal, R., Trejo, L.J. & Matthews, B. 2003. Kernel PLS-SVC for Linear and Nonlinear Classification. In *Proceedings, Twentieth International Conference on Machine Learning*. 640–647.

S. De Jong. 1993. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems*. 18:251–263.

Schwarte, A. & Isermann, R. 2002. Neural network applications for model based fault detection with parity equations. *IFAC Proceedings Volumes*. 35(1):205–210.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A. & Doyle, F.J. 2010. *Process Dynamics and Control*. 3rd ed. Hoboken, N.J.: John Wiley & Sons.

Shlens, J. 2014. A Tutorial on Principal Component Analysis. [Online], Available: http://arxiv.org/abs/1404.1100.

Steinwart, I. & Christmann, A. 2008. *Support Vector Machines*. 1st ed. New York, N.Y.: Springer-Verlag.

Stone, M. 1974. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*. 36(2):111–133.

von Stosch, M., Oliveira, R., Peres, J. & Feyo de Azevedo, S. 2014. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Computers and Chemical Engineering*. 60:86–101.

Vapnik, V. 1998. *Statistical Learning Theory*. 1st ed. New York: Wiley.

Vapnik, V., Golowich, S.E. & Smola, A. 1997. Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems*. 281–287.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. & Yin, K. 2003. A Review of Process Fault Detection and Diagnosis Part III: Process History Based Methods. *Computers & Chemical Engineering*. 27:327–346.

Wahba, G. 1992. Multivariate Function and Operator Estimation, Based on Smoothing Splines

and Reproducing Kernels. In Vol. 12. Addison-Weasley *Non-linear Modeling and Forecasting, SFI Studies in the Sciences of Complexity, Proc Vol XII*. 95–112.

Wierda, S.J. 1994. Multivariate statistical process control—recent results and directions for future research. *Statistica Neerlandica*. 48(2):147–168.

Willis, M.J. & von Stosch, M. 2017. Simultaneous parameter identification and discrimination of the nonparametric structure of hybrid semi-parametric models. *Computers and Chemical Engineering*. 104:366–376.

Wold, S., Ruhe, A., Wold, H. & Dunn III, W.J. 1984. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*. 5(3):735–743.

Wold, S., Esbensen, K. & Geladi, P. 1987. Principal Component Analysis. *Chemometrics and intelligent laboratory systems*. 2(1–3):37–52.

Yekkehkhany, B., Safari, A., Homayouni, S. & Hasanlou, M. 2014. A comparison study of different kernel functions for SVM-based classification of multi-temporal polarimetry SAR data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. 40(2W3):281–285.

Yin, S., Gao, X., Karimi, H.R. & Zhu, X. 2014. Study on support vector machine-based fault detection in Tennessee Eastman process. *Abstract and Applied Analysis*. 2014.

APPENDIX A – SIMULINK MODEL IMPLEMENTATION

A Simulink model of the non-isothermal jacketed CSTR model proposed in Section 4.1 was developed to generate process datasets on which the fault identification models will be trained. The overall Simulink model of the CSTR is shown in Figure 61 below. Feedback control is implemented in the model to maintain the outlet concentration by adjusting the inlet flowrate. The controller is implemented through the PID controller block with control parameters set as the values listed in Table 4 above.

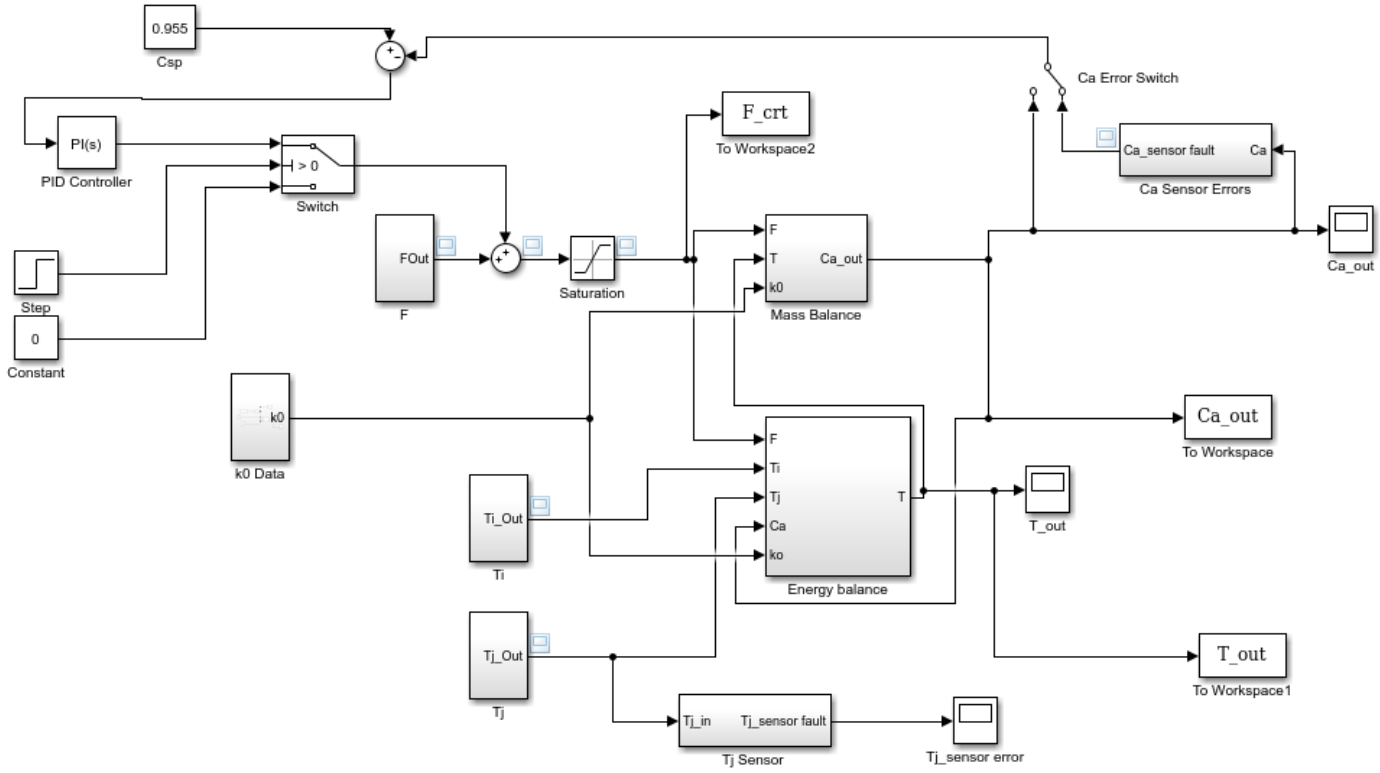

Figure 61: Simulink model of CSTR

Two subsystems were created to contain the material balance and the energy balance. The material balance of A (Equation *45*) is implemented in the mass balance subsystem as seen in Figure 62.

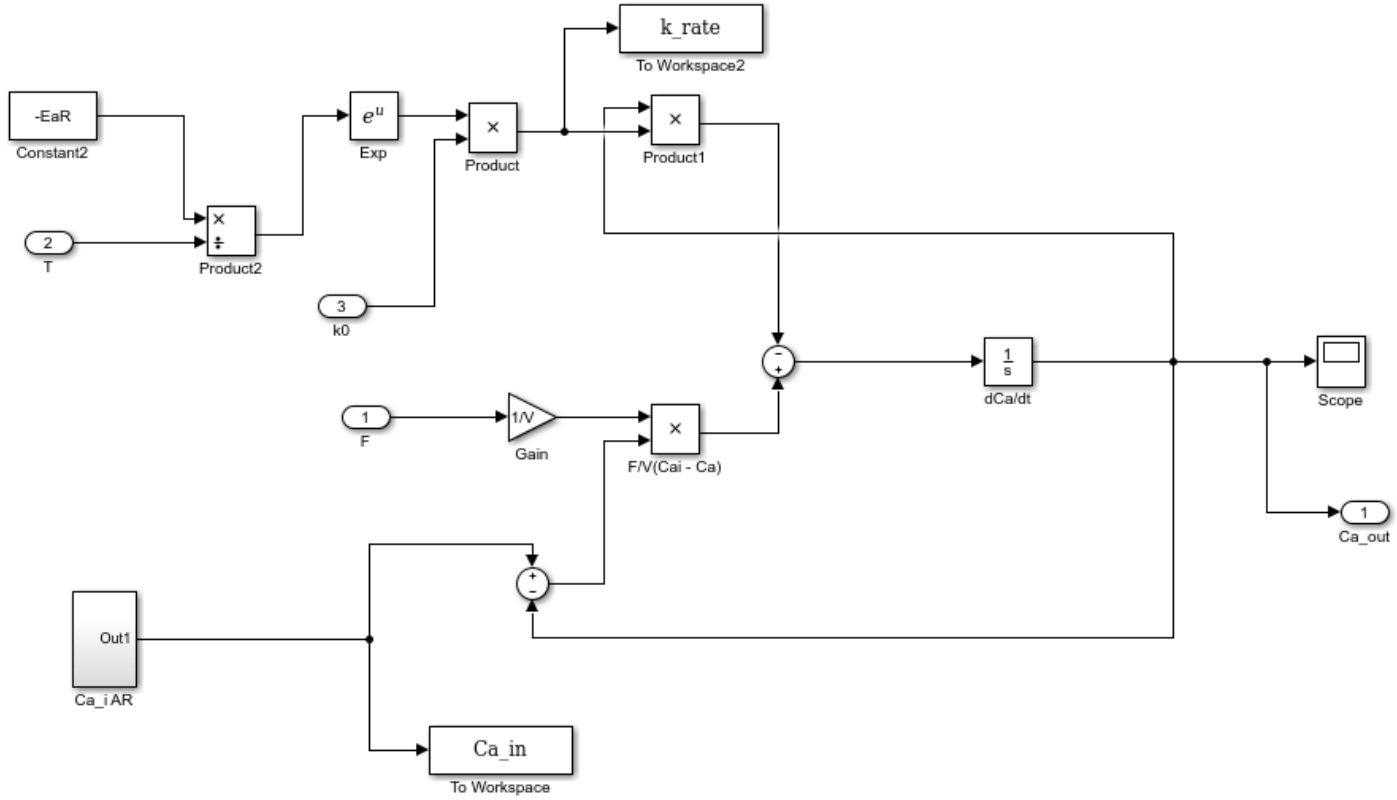

Figure 62: Material balance of A in the mass balance subsystem

The energy balance of the CSTR (Equation 46) is implemented in the energy balance subsystem of the Simulink model as shown in Figure 63.

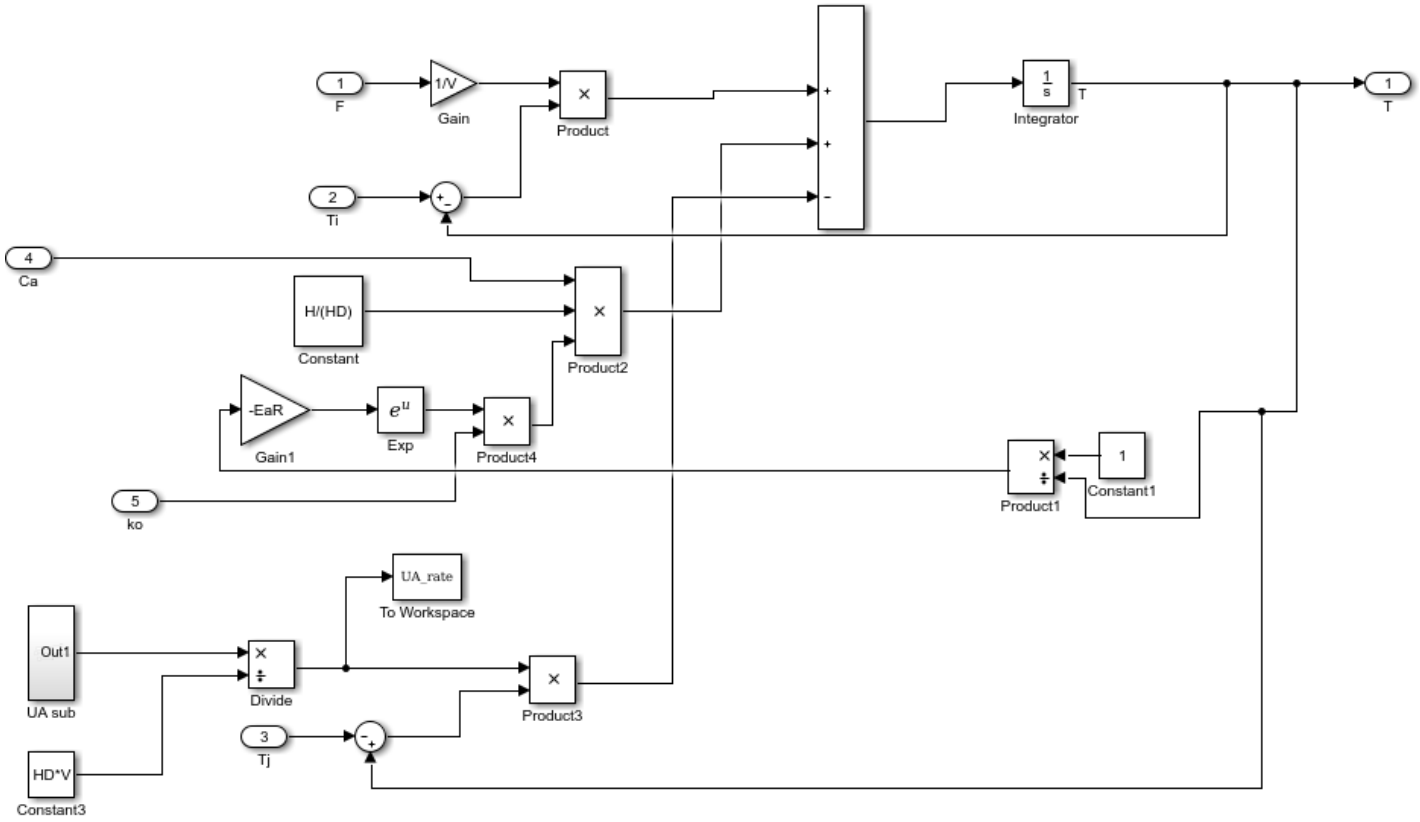

Figure 63: Energy balance subsystem

Each process input has noise associated with the readings; the noise is used to cause the Simulink model to more closely resemble a real world process. The input noise is modelled as a first-order autoregressive process as shown in Equation 51 where ϕ represents the autoregressive constant and e the normally distributed noise ($e \sim N(0, \sigma_e^2)$). For this model the value of ϕ is set as 0.9. The noise variance for each variable is captured in Table 10.

$$y_{i+1} = y_i \phi + e \qquad [\ 51\ ]$$

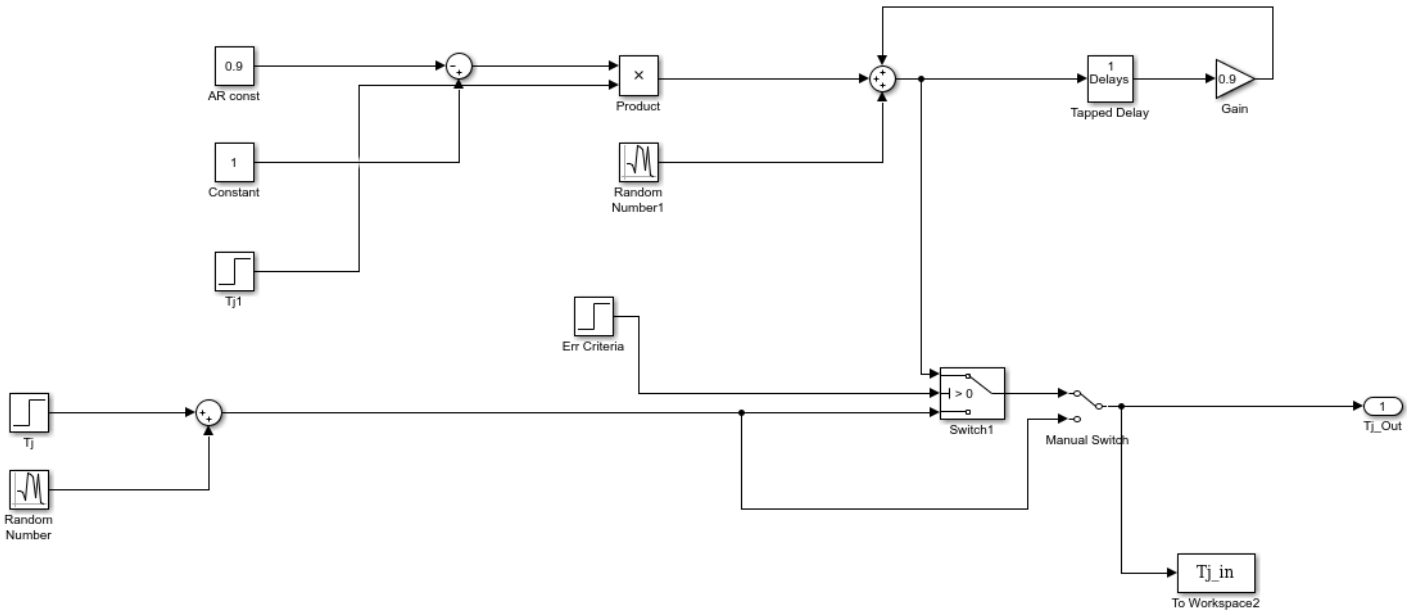

Figure 64: Input fault application to T_J

Table 10 : Variable noise variances

VARIABLE	VARIANCE (σ_e^2)
F	1.9
T_I	0.475
T_J	0.05

Sensor error is applied to $C_{a,out}$ to mimic real world conditions. The error is comprised of sensor drift applied to the concentration reading. The sensor drift is modelled a ramp signal which is added to the true concentration. This drift error is modelled in Simulink as seen in Figure 65 below. The white noise added to the concentration is normally distributed with zero with a variance of 0.001.

Figure 65: Sensor error applied to Ca_{out}

The process faults applied to model take the form ramp input with the gradient of the ramp set as the magnitude of the error. The catalyst deactivation fault implementation in Simulink is shown in Figure 66. The fault ramp was determined using a MATLAB script where the fault was randomly implemented using the durations and variances described in Table 5 and Table 6 and saved to a workspace named k0Data.mat. The workspace is then read into Simulink and the fault is applied. A saturation block used to ensure that the k0 fault does not result in a negative k0 parameter.

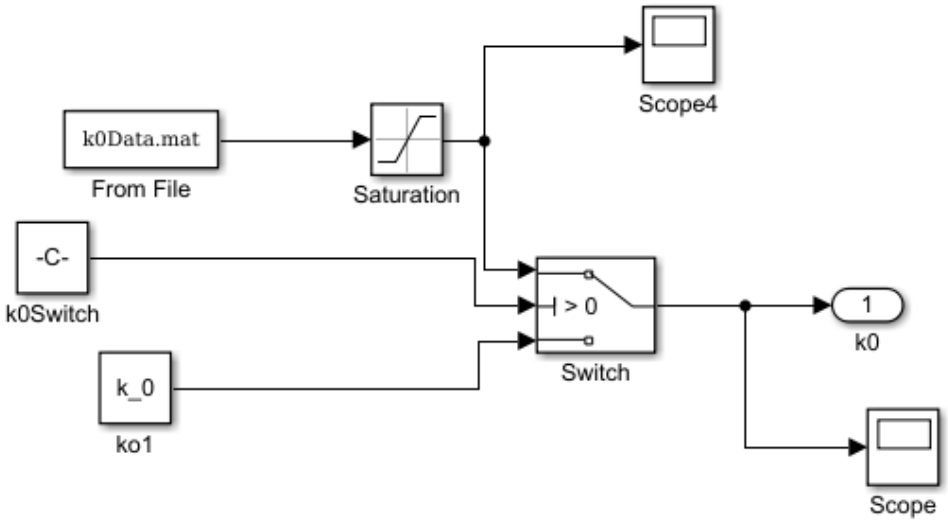

Figure 66: Catalyst deactivation fault Simulink implementation

APPENDIX B – NUMERICAL VALUES FOR MAPE, SENSITIVITIES AND SPECIFICITIES

The following appendix contains the numerical values for the MAPE, sensitivities and specificities used in the plots in the Chapter 4. The training data sets used for sensitivity and specificity comparison plots were subjected to k-fold cross-validation where k was set as 10. For each cross-validation case (or fold) the respective model was trained using the training data and the sensitivities and specificities were calculated using the testing data for the respective cross validation case.

Table 11: MAPE values for dynamic PLS window lengths

Window span (min)	Cross Validation Cases									
	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10
1	0.2600	0.2612	0.2658	0.2665	0.2651	0.2651	0.2631	0.2671	0.2667	0.2641
2	0.2594	0.2589	0.2633	0.2646	0.2622	0.2628	0.2618	0.2651	0.2644	0.2614
3	0.2586	0.2588	0.2613	0.2633	0.2609	0.2602	0.2606	0.2624	0.2618	0.2583
4	0.2573	0.2576	0.2596	0.2613	0.2603	0.2595	0.2601	0.2609	0.2602	0.2564
5	0.2560	0.2562	0.2581	0.2604	0.2587	0.2589	0.2592	0.2613	0.2591	0.2550
6	0.2552	0.2552	0.2572	0.2600	0.2583	0.2595	0.2585	0.2614	0.2591	0.2540
7	0.2548	0.2549	0.2566	0.2598	0.2582	0.2597	0.2587	0.2611	0.2592	0.2541
8	0.2547	0.2559	0.2555	0.2572	0.2581	0.2596	0.2600	0.2616	0.2598	0.2547
9	0.2540	0.2548	0.2559	0.2597	0.2562	0.2611	0.2593	0.2623	0.2597	0.2546
10	0.2550	0.2560	0.2582	0.2588	0.2573	0.2617	0.2591	0.2628	0.2594	0.2555
Average	0.2565	0.2569	0.2591	0.2612	0.2595	0.2608	0.2600	0.2626	0.2609	0.2568

The sensitivity and specificity used in the respective sensitivity and specificity curves were obtained by first obtaining model data at each process fault magnitude considered. The process data was then partitioned into a training dataset and a test dataset with the test set containing 10% of the available model data. The fault identification models were trained using the training data and the sensitivity and specificity were then calculated using the test data as described in Chapter 2.8.

Table 12: Sensitivity curve values for catalyst deactivation fault

Magnitude	Kernel SVM	Standard PLS	Dynamic PLS
6.94	0.7550	0.7414	0.7826
8.33	0.7179	0.7485	0.7953
9.72	0.7407	0.7469	0.7864
11.11	0.7136	0.7670	0.8164
12.50	0.7745	0.7824	0.8252
13.89	0.8263	0.7606	0.8205
15.28	0.7887	0.7537	0.8191
16.67	0.8093	0.8062	0.8449
18.06	0.8088	0.7984	0.8353
19.44	0.8293	0.7967	0.8424
20.83	0.8421	0.7956	0.8419
22.22	0.8142	0.8105	0.8508
23.61	0.8750	0.8232	0.8547
25.00	0.7557	0.7864	0.8278
26.39	0.8241	0.8169	0.8618

27.78	0.8033	0.8149	0.8707
29.17	0.8376	0.8188	0.8596
30.56	0.7477	0.8286	0.8688
31.94	0.8438	0.8192	0.8590
33.33	0.8042	0.8215	0.8569
34.72	0.8768	0.8142	0.8688
36.11	0.7926	0.8268	0.8716
37.50	0.8040	0.8242	0.8757
38.89	0.8626	0.8435	0.8880
40.28	0.7811	0.8210	0.8554
41.67	0.8586	0.8488	0.8880
43.06	0.8396	0.8322	0.8765
44.44	0.8177	0.8387	0.8844
45.83	0.8284	0.8284	0.8749
47.22	0.8186	0.8413	0.8783
48.61	0.8333	0.8440	0.8844
50.00	0.7970	0.8181	0.8739
51.39	0.8513	0.8372	0.8832
52.78	0.7981	0.8542	0.8898
54.17	0.8299	0.8221	0.8700
55.56	0.8358	0.8444	0.8848
56.94	0.8438	0.8584	0.9003

58.33	0.8116	0.8490	0.8938
59.72	0.8164	0.8428	0.8928
61.11	0.8534	0.8429	0.8988
62.50	0.8492	0.8598	0.8745

Table 13: Specificity curve values for catalyst deactivation fault

Magnitude	Kernel SVM	Standard PLS	Dynamic PLS
6.94	0.9915	0.9931	0.9969
8.33	0.9908	0.9892	0.9985
9.72	0.9878	0.9925	0.9969
11.11	0.9945	0.9939	0.9992
12.50	0.9931	0.9876	0.9992
13.89	0.9922	0.9961	0.9977
15.28	0.9900	0.9869	0.9969
16.67	0.9908	0.9946	0.9992
18.06	0.9931	0.9916	0.9969
19.44	0.9918	0.9901	0.9977
20.83	0.9924	0.9962	0.9992
22.22	0.9916	0.9900	0.9969
23.61	0.9930	0.9969	0.9985
25.00	0.9940	0.9939	0.9955
26.39	0.9931	0.9922	0.9985
27.78	0.9916	0.9901	1.0000
29.17	0.9900	0.9883	0.9992
30.56	0.9906	0.9938	0.9984
31.94	0.9946	0.9931	0.9969

33.33	0.9924	0.9939	1.0000
34.72	0.9876	0.9924	0.9961
36.11	0.9924	0.9915	0.9961
37.50	0.9892	0.9915	0.9969
38.89	0.9915	0.9916	0.9985
40.28	0.9908	0.9884	0.9985
41.67	0.9923	0.9939	0.9969
43.06	0.9953	0.9922	0.9954
44.44	0.9961	0.9931	0.9984
45.83	0.9910	0.9924	0.9985
47.22	0.9931	0.9876	0.9970
48.61	0.9901	0.9877	0.9992
50.00	0.9916	0.9862	0.9984
51.39	0.9954	0.9915	0.9985
52.78	0.9923	0.9893	0.9970
54.17	0.9939	0.9962	0.9977
55.56	0.9892	0.9916	0.9977
56.94	0.9901	0.9923	0.9985
58.33	0.9915	0.9954	0.9977
59.72	0.9907	0.9931	0.9985
61.11	0.9862	0.9901	0.9969
62.50	0.9915	0.9901	0.9977

Table 14: Sensitivity values for catalyst deactivation-inlet fault trained model with catalyst deactivation fault active

CV Cases	kSVM	r-kSVM	PLS-kSVM	dPLS-kSVM	SVM	r-SVM	PLS-SVM	dPLS-SVM
CV1	0.8492	0.9202	0.8698	0.8689	0.8241	0.9468	0.8281	0.9468
CV2	0.8482	0.8667	0.8557	0.8702	0.8429	0.8974	0.8497	0.8974
CV3	0.8085	0.8601	0.8081	0.8857	0.8191	0.9016	0.8673	0.8964
CV4	0.8883	0.8724	0.8564	0.8592	0.8511	0.8673	0.8367	0.8673
CV5	0.8507	0.8838	0.8482	0.8619	0.8109	0.9091	0.8220	0.9091
CV6	0.8421	0.8848	0.8737	0.8706	0.8316	0.8848	0.8220	0.8848
CV7	0.9048	0.9021	0.8723	0.8768	0.8889	0.9381	0.8177	0.9330
CV8	0.8750	0.8854	0.8507	0.8384	0.8438	0.8958	0.8168	0.8958
CV9	0.8281	0.8883	0.8852	0.8622	0.8281	0.8777	0.8505	0.8777
CV10	0.8274	0.9096	0.8783	0.8945	0.8173	0.9309	0.8307	0.9309
Average	0.8522	0.8873	0.8598	0.8688	0.8358	0.9050	0.8342	0.9039
Std. dev	0.0295	0.0189	0.0220	0.0154	0.0227	0.0264	0.0168	0.0258

Table 15: Specificity values for catalyst deactivation-inlet concentration fault trained model with catalyst deactivation fault active

CV Cases	kSVM	r-kSVM	PLS-kSVM	dPLS-kSVM	SVM	r-SVM	PLS-SVM	dPLS-SVM
CV1	0.9944	0.9977	0.9944	0.9944	0.9860	0.9954	0.9860	0.9831
CV2	0.9830	0.9985	0.9830	0.9829	0.9943	0.9969	0.9972	0.9943
CV3	0.9916	0.9985	0.9944	0.9944	0.9944	0.9954	0.9860	0.9888
CV4	0.9831	0.9962	0.9831	0.9830	0.9774	0.9962	0.9802	0.9802
CV5	0.9860	0.9992	0.9860	0.9860	0.9916	0.9946	0.9888	0.9916
CV6	0.9915	0.9977	0.9915	0.9915	0.9887	0.9969	0.9887	0.9887
CV7	0.9860	0.9977	0.9860	0.9860	0.9888	0.9931	0.9916	0.9888
CV8	0.9971	1.0000	0.9971	0.9971	0.9826	0.9977	0.9884	0.9855
CV9	0.9915	0.9977	0.9915	0.9915	0.9887	0.9954	0.9887	0.9859
CV10	0.9944	0.9985	0.9944	0.9944	0.9860	0.9969	0.9888	0.9860
Average	0.9899	0.9982	0.9901	0.9901	0.9879	0.9959	0.9884	0.9873
Std. dev	0.0050	0.0010	0.0052	0.0052	0.0052	0.0014	0.0043	0.0041

Table 16: Sensitivity values for catalyst deactivation-overall heat transfer coefficient fault trained model with heat transfer fault active

CV Cases	kSVM	r-kSVM	PLS-kSVM	dPLS-kSVM	SVM	r-SVM	PLS-SVM	dPLS-SVM
CV1	0.7677	0.7742	0.7548	0.7219	0.7677	0.8645	0.7677	0.6821
CV2	0.6420	0.6790	0.6728	0.7222	0.6420	0.7901	0.6420	0.7593
CV3	0.7750	0.7750	0.7688	0.7250	0.7750	0.8375	0.7750	0.8000
CV4	0.7919	0.7584	0.7248	0.7436	0.7919	0.8188	0.7919	0.8718
CV5	0.6943	0.7643	0.7197	0.6548	0.6943	0.8025	0.7006	0.6726
CV6	0.8383	0.7186	0.7006	0.7793	0.8383	0.7844	0.8323	0.7724
CV7	0.6623	0.7208	0.6688	0.6832	0.6623	0.7987	0.6623	0.7578
CV8	0.7329	0.7081	0.6832	0.7261	0.7329	0.8199	0.7329	0.7643
CV9	0.8182	0.7013	0.7078	0.7818	0.8182	0.7532	0.8182	0.7758
CV10	0.7953	0.7368	0.7193	0.7452	0.7953	0.8363	0.7895	0.7580
Average	0.7518	0.7336	0.7121	0.7283	0.7518	0.8106	0.7513	0.7614
Std. dev	0.0666	0.0333	0.0328	0.0388	0.0666	0.0317	0.0648	0.0560

Table 17: Sensitivity values for catalyst deactivation-overall heat transfer coefficient fault trained model with catalyst deactivation fault active

CV Cases	kSVM	r-kSVM	PLS-kSVM	dPLS-kSVM	SVM	r-SVM	PLS-SVM	dPLS-SVM
CV1	0.7000	0.6000	0.6400	0.6400	0.7000	0.7000	0.7000	0.7000
CV2	0.6522	0.5870	0.5870	0.5870	0.6522	0.6522	0.6522	0.6957
CV3	0.6429	0.6190	0.6190	0.6190	0.6429	0.6429	0.6429	0.6905
CV4	0.7917	0.6250	0.6458	0.6458	0.7917	0.7917	0.7917	0.7917
CV5	0.6875	0.6667	0.6875	0.6875	0.6875	0.6875	0.6875	0.6875
CV6	0.6667	0.5208	0.5417	0.5625	0.6667	0.6667	0.6667	0.6875
CV7	0.6667	0.7333	0.7333	0.7556	0.6667	0.6667	0.6667	0.6444
CV8	0.6364	0.6818	0.6818	0.6818	0.6364	0.6591	0.6364	0.6591
CV9	0.6905	0.6667	0.6429	0.6429	0.6905	0.6905	0.6905	0.6667
CV10	0.7037	0.6296	0.6296	0.6296	0.7037	0.7037	0.7037	0.7222
Average	0.6838	0.6330	0.6409	0.6452	0.6838	0.6861	0.6838	0.6945
Std. dev	0.0446	0.0585	0.0535	0.0543	0.0446	0.0424	0.0446	0.0407

Table 18: Specificity values for catalyst deactivation-overall heat transfer coefficient fault trained model with heat transfer fault active

CV Cases	kSVM	r-kSVM	PLS-kSVM	dPLS-kSVM	SVM	r-SVM	PLS-SVM	dPLS-SVM
CV1	0.9549	0.9631	0.9590	0.9440	0.9918	0.9959	0.9918	0.9549
CV2	0.9202	0.9370	0.9202	0.9447	0.9664	0.9790	0.9748	0.9202
CV3	0.9292	0.9375	0.9292	0.9697	0.9667	0.9792	0.9667	0.9292
CV4	0.8720	0.8880	0.8720	0.9449	0.9040	0.9240	0.9000	0.8720
CV5	0.9215	0.9339	0.9215	0.9341	0.9752	0.9835	0.9752	0.9215
CV6	0.8707	0.8836	0.8707	0.9339	0.8664	0.9440	0.8664	0.8707
CV7	0.9796	0.9755	0.9796	0.9541	0.9878	0.9878	0.9878	0.9796
CV8	0.9538	0.9622	0.9538	0.9440	0.9664	0.9916	0.9706	0.9538
CV9	0.8408	0.8694	0.8408	0.9268	0.8204	0.9265	0.8163	0.8408
CV10	0.8860	0.8904	0.8860	0.9352	0.8991	0.9693	0.8991	0.8860
Average	0.9129	0.9241	0.9133	0.9431	0.9344	0.9681	0.9349	0.9129
Std. dev	0.0443	0.0382	0.0447	0.0122	0.0584	0.0268	0.0604	0.0443

Table 19: Specificity values for catalyst deactivation-overall heat transfer coefficient fault trained model with catalyst deactivation fault active

CV Cases	kSVM	r-kSVM	PLS-kSVM	dPLS-kSVM	SVM	r-SVM	PLS-SVM	dPLS-SVM
CV1	0.9857	0.9857	0.9857	0.9857	0.9943	0.9943	0.9943	0.9943
CV2	0.9859	0.9859	0.9859	0.9859	0.9915	0.9915	0.9887	0.9887
CV3	0.9888	0.9888	0.9888	0.9888	0.9860	0.9860	0.9832	0.9832
CV4	0.9886	0.9886	0.9886	0.9886	0.9943	0.9943	0.9915	0.9915
CV5	0.9858	0.9858	0.9858	0.9858	0.9829	0.9829	0.9829	0.9829
CV6	0.9801	0.9829	0.9829	0.9829	0.9858	0.9858	0.9858	0.9858
CV7	0.9746	0.9774	0.9746	0.9746	0.9887	0.9887	0.9887	0.9887
CV8	0.9859	0.9859	0.9859	0.9859	0.9831	0.9831	0.9831	0.9831
CV9	0.9776	0.9776	0.9776	0.9776	0.9972	0.9972	0.9972	0.9972
CV10	0.9855	0.9855	0.9855	0.9855	0.9855	0.9855	0.9855	0.9855
Average	0.9838	0.9844	0.9841	0.9841	0.9889	0.9889	0.9881	0.9881
Std. dev	0.0048	0.0040	0.0046	0.0046	0.0051	0.0051	0.0050	0.0050

Table 20: Sensitivity curve values for heat transfer fault

Fault Magnitude	0.15	0.152	0.154	0.156	0.158	0.16
SVM	0.7100	0.6736	0.7055	0.7046	0.6919	0.7518
PLS-SVM	0.7100	0.6736	0.7055	0.7046	0.6919	0.7513
dPLS-SVM	0.7153	0.6719	0.7128	0.7161	0.7018	0.7614

Table 21: Specificity curve values for heat transfer fault

Fault Magnitude	0.15	0.152	0.154	0.156	0.158	0.16
SVM	0.9627	0.9631	0.9626	0.9523	0.9695	0.9344
PLS-SVM	0.9623	0.9635	0.9626	0.9523	0.9695	0.9349
dPLS-SVM	0.9675	0.9689	0.9677	0.9541	0.9659	0.9497