Improved Slow Feature Analysis for Process Monitoring

by

Hussein Saafan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Chemical Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis consists in part of two manuscripts written for publication, which are co-authored by my supervisor and I:

Citations:

[1] Saafan, H., & Zhu, Q. (2022). Improved Manifold Sparse Slow Feature Analysis for Process Monitoring. Computers & Chemical Engineering, 107905.

[2] Saafan, H., & Zhu, Q. (2021, May). Comprehensive Monitoring with Incremental Slow Feature Analysis. In 2021 American Control Conference (ACC) (pp. 917-922). IEEE.

The detailed authorship contributions are

Hussein Saafan: Co-development and implementation of the MSSFA algorithm and framework in [1] and the IncSFA based monitoring scheme in [2]. Writing and editing of the manuscripts for [1] and [2].

Qinqin Zhu: Co-development of the MSSFA algorithm and framework in [1] and the IncSFA based monitoring scheme in [2]. Writing and editing of the manuscripts for [1] and [2].

Abstract

Unsupervised multivariate statistical analysis models are valuable tools for process monitoring and fault diagnosis. Among them, slow feature analysis (SFA) is widely studied and used due to its explicit statistical properties, which aims to extract invariant features of temporally varying signals. This inclusion of dynamics in the model is important when working with process data where new samples are highly correlated to previous ones. However, the existing variations of SFA models cannot exploit increasingly tremendous data volume in modern industries, since they require the data to be fed in as a whole in the training stage. Further, sparsity is also desirable to provide interpretable models and prevent model overfitting.

To address the aforementioned issues, a novel algorithm for inducing sparsity in SFA is first introduced, which is referred to as manifold sparse SFA (MSSFA). The non-smooth sparse SFA objective function is optimized using proximal gradient descent and the SFA constraint is fulfilled using manifold optimization. An associated fault detection and diagnosis framework is developed that retains the unsupervised nature of SFA. When compared to SFA, sparse SFA (SSFA), and sparse principal component analysis (SPCA), MSSFA shows superior performance in computational complexity, interpretability, fault detection, and fault diagnosis on the Tennessee Eastman process (TEP) and three-phase flow facility (TPFF) data sets. Furthermore, its sparsity is much improved over SFA and SSFA.

Further, to exploit the increasing number of collected samples efficiently, a covariance free incremental SFA (IncSFA) is adapted in this work, which handles massive data efficiently and has a linear feature updating complexity with respect to data dimensionality. The IncSFA based process monitoring scheme is also proposed for anomaly detection. Further, a new incremental MSSFA (IncMSSFA) algorithm is also introduced that is able to use the same monitoring scheme. These two algorithms are compared against recursive SFA (RSFA) which can also process data incrementally. The efficiency of IncSFA-based monitoring is demonstrated with the TEP and TPFF data sets. The inclusion of sparsity in the IncMSSFA method provides superior monitoring performance at the cost of a quadratic complexity in terms of data dimensionality. This complexity is still an improvement over the cubic complexity of RSFA.

Acknowledgements

Firstly, I would like to thank my supervisor, Professor Zhu, whose guidance and expertise has been invaluable to me. I would also like to thank and acknowledge NSERC and the University of Waterloo for funding my research activities. Finally, I would like to thank my friends and family for giving me the strength to succeed in my education.

Dedication

This thesis is dedicated to my family that supported me throughout my last six years of post secondary education as well as the eighteen years before that.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AIC Akaike information criterion 16

APGD accelerated proximal gradient descent 10, 21, 22, 81

CCA canonical-correlation analysis 3

CCIPCA candid covariance-free incremental principal component analysis 52, 53, 55, 59, 61

CDC complete decomposition contribution 25, 26, 38, 67

CIMCA covariance-free incremental minor component analysis 52, 53, 61

df degrees of freedom 6

FAR false alarm rate 30, 43, 45, 64, 72, 74

FDR fault detection rate 30, 33, 43, 45, 64, 72, 74

FISTA fast iterative soft thresholding algorithm 10, 29

ICA independent component analysis 1, 3

IncMSSFA incremental manifold sparse slow feature analysis 4, 57, 59–61, 64, 67, 72, 74, 77, 80–82

IncSFA incremental slow feature analysis 3, 4, 9, 52, 53, 55, 57, 59–61, 64, 67, 72, 74, 80, 81

ISTA iterative soft thresholding algorithm 10

MSSFA manifold sparse slow feature analysis 4, 21, 22, 24, 26, 29, 30, 38, 45, 48, 51, 57, 80–82, 89, 90

PCA principal component analysis 1–3, 16, 26

PDF probability density function 5

PGD proximal gradient descent 10

PLS partial least squares 1

RSFA recursive slow feature analysis 3, 4, 9, 55, 57, 59–61, 64, 67, 72, 74, 80, 81

SFA slow feature analysis 3, 4, 7–9, 12, 13, 21, 26, 28–30, 33, 38, 39, 45, 48, 51, 53, 80, 81

SPCA sparse principal component analysis 2, 4, 13, 26, 28–30, 33, 45, 51, 81

SPE squared prediction error 13, 33

SSFA sparse slow feature analysis 3, 4, 12, 21, 26, 28–30, 38, 39, 45, 48, 51, 81, 90

SVD singular value decomposition 8, 9, 29, 53, 59–61, 64, 67, 72, 74

TEP Tennessee Eastman process 4, 14, 18, 26, 29, 30, 39, 51, 57, 61, 80, 81, 90

TPFF three-phase flow facility 4, 18, 43, 45, 51, 72, 80, 81

Chapter 1

Introduction

Technology has always shaped society, from the Agricultural Revolution 10000 years ago that turned hunters into farmers [1] to the First Industrial Revolution 260 years ago that signalled the end of feudalism. There have been two other industrial revolutions since that time. The second was powered by oil and electricity and led to automobiles and mass production. The third introduced electronics and increased automation. The Fourth Industrial Revolution is taking place right now and bringing with it technologies such as green energy, 3D printing, and genetic engineering [2].

This revolution has brought with it an immense amount of data which has been invaluable in its contribution to artificial intelligence. In the context of process control, this data comes in the form of sensor readings and could be used to make inferences about quality, efficiency, and safety. These new capabilities have the potential to increase the resource efficiency and flexibility of manufacturing processes as well as allowing for individualization on demand.

1.1 Process Data

Process data presents a series of requirements for any analysis technique. The first of these is that the technique should consider the time variance of data. In processes, the current data sample can be highly dependent on the previous samples. Many methods such as principal component analysis (PCA) [3], partial least squares (PLS) [4], and independent component analysis (ICA) [5] make the implicit assumption that data is static and independent in temporal scale, which is not valid for most industrial processes.

The technique should also be able to adapt to new scenarios. It is unlikely that data representing all operating conditions is available. Faults can occur for all sorts of reasons and gathering ample data for each of these faults could be costly or infeasible. In addition to this, the normal operating conditions can change due to changes to the product, equipment, or environment. This might also affect how the data changes when presented with faults.

The actions and conclusions of the technique should also be explainable to humans. Industrial environments, especially chemical plants, have the potential to be great hazards when operated improperly. For quality optimization, a qualified engineer should be able to understand why the technique wants to make a certain change before giving final approval. In fault detection, the engineer should be able to understand why the technique detected a fault and where the fault has occurred.

Another possible requirement is computational and storage efficiency. If a technique requires the storage of large amounts of historical data, it may not be feasible for many industrial environments. In addition to this, the actions and conclusions of a technique may be time sensitive. More complex techniques would require the use of more expensive hardware to meet this need.

1.2 Current Process Monitoring Techniques

Broadly speaking, process monitoring techniques can be split into two categories, supervised and unsupervised. Supervised techniques require additional information about training data. This could come in the form of labels that indicate when a fault has occurred or labels that indicate exactly which fault is occurring. Unsupervised techniques do not require any labelled data which is desirable when working with process data as gathering data about every operating condition or fault that may arise would prove intractable.

One of the most popular unsupervised techniques is PCA. PCA finds a set of orthogonal vectors with each successive vector pointing in the direction of maximal variance once the effect of the previous vectors has been removed from the data [3]. This can be viewed as a linear transformation, and a reduction of the number of dimensions of the data can be performed by only keeping the first few columns of the transformation matrix [6].

There are many extensions to PCA, one which is important to process data monitoring is dynamic PCA which augments the original data by appending copies of the previous samples to the current sample [6]. This technique allows for the incorporation of dynamic relationships into PCA. The same technique can be applied to other algorithms by passing them the augmented data matrix as the input. Another extension is sparse principal

component analysis (SPCA) which adds an objective of maximizing the number of 0 values in the transformation matrix [7]. This allows for the transformation to be understood by humans which is important in process monitoring where the input signals have physical meaning. PCA and dynamic PCA based techniques have been used extensively for process monitoring applications such as in [8, 9, 10, 11, 12].

Other techniques include ICA and canonical-correlation analysis (CCA). ICA is related to PCA except that instead of finding vectors in the direction of maximal variance, the vectors are chosen such that their statistical dependence is minimized [5]. CCA aims to find linear relationships between two sets of variables [13]. These models have been studied in the context of process monitoring such as in [14, 15, 16] for ICA and [17, 18, 19] for CCA.

The focus of this thesis will be on another technique called slow feature analysis (SFA). SFA is set apart from these methods by the inclusion of process dynamics in its formulation. SFA finds a transformation of input signals to features that minimizes the speed (i.e. the sum of squares of its first derivative) of the features [20]. Additional modifications to SFA exist to tackle different problems. Algorithms such as incremental slow feature analysis (IncSFA) [21] and recursive slow feature analysis (RSFA) [22] are able to update the model with single sample inputs. This is useful for online training as when a new normal operating condition is encountered, the model can update itself rather than requiring retraining. This also prevents the need for keeping large stores of historical data in memory or storage. Another modification is sparse slow feature analysis (SSFA) [23] which creates sparse transformation matrices. Some works that use SFA for process monitoring include [22, 23, 24, 25].

Most of the unsupervised process monitoring works mentioned here will make use of statistical distributions such as the Hotelling T^2 distribution [26] along with their associated control limits to label data as normal or abnormal. Going one step further, some works will introduce a fault diagnosis model which will provide information about where or why the fault occurred. The unsupervised approaches presented in [27] will assign blame to the input signals that had the highest contribution to the monitoring statistics.

1.3 Research Outcomes

A framework that fulfills the requirements of process data is desired. This framework would include process dynamics, be able to make incremental updates, and provide a sparse model of the process.

A novel sparse algorithm called manifold sparse slow feature analysis (MSSFA) is developed with a corresponding monitoring and diagnosis framework. This algorithm makes use of proximal algorithms to induce sparsity. It also uses manifold optimization to fulfill the constraints of SFA based algorithms and efficiently arrive at a solution. This algorithm provides superior computational complexity, interpretability, fault detection, and fault diagnosis performance when compared with SFA, SSFA, and SPCA using the Tennessee Eastman process (TEP) and three-phase flow facility (TPFF) data sets.

After this, a process monitoring and fault diagnosis framework is developed for use with IncSFA. This scheme is meant to exploit the superior computational complexity of IncSFA for process monitoring applications. Another incremental version of MSSFA called incremental manifold sparse slow feature analysis (IncMSSFA) is then developed. The monitoring and diagnosis framework that was developed for IncSFA is used here to great effect. IncMSSFA is the only SFA algorithm that is both incremental and sparse. This algorithm and framework is able to fulfill all the process data requirements outlined above. IncMSSFA had better interpretability, fault detection, and fault diagnosis performance than both RSFA and IncSFA using the TEP and TPFF sets. It could not achieve the linear complexity of IncSFA, but its quadratic complexity was lower than the cubic complexity of RSFA.

1.4 Thesis Outline

The rest of this thesis is outlined as follows. Chapter 2 provides a background on some selected topics that are important to the rest of the thesis. After this, Chapter 3 describes the MSSFA algorithm in detail, provides a corresponding monitoring scheme, and an investigation into its performance when compared with other sparse algorithms. Chapter 4 details the incremental process monitoring scheme that was developed for IncSFA and introduces the new IncMSSFA algorithm. These algorithms, along with RSFA, are compared to determine which incremental algorithm is better for process monitoring. Finally, Chapter 5 sums up the findings and provides some recommendations for future work.

Chapter 2

Background Information

2.1 Mathematical Background

It is assumed that the reader has at least an introductory background in both statistics and linear algebra. The following section will cover some specific concepts that are important for the remainder of this thesis.

Covariance measures how two variables interact with each other. The covariance is positive if there is a trend in which both variables increase or decrease in tandem, and it is negative if there is a trend in which increases in one variable are observed with decreases in the other. It is useful to be able to represent the covariance of all variables in a matrix. For a data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ that consists of n sample columns each containing m variables, the covariance matrix is calculated using Equation (2.1) [28].

$$\text{Cov}(\mathbf{X}) = \frac{1}{n-1}\mathbf{X}\mathbf{X}^\top \tag{2.1}$$

This matrix is symmetric and each entry $\text{Cov}(\mathbf{X})_{i,j}$ is the covariance between the ith and jth variable. The variance of each variable is found along the diagonal. The covariance matrix is also positive semi-definite [28] which is defined in Equation (2.2).

$$\mathbf{M} \text{ is Positive Semi-Definite} \Leftrightarrow \mathbf{x}^\top \mathbf{M}\mathbf{x} \geq 0 \quad \forall\, \mathbf{x} \in \mathbb{R}^m \tag{2.2}$$

A Gaussian distribution is a continuous statistical distribution whose probability density function (PDF) takes the form of Equation (2.3) where μ is the mean and σ^2 is the

variance [29].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{2.3}$$

The standard normal distribution is a Gaussian distribution that has a mean of 0 and a variance of 1 [29].

The distribution of the sum of squares of k random variables that are each distributed according to a standard normal distribution is called the χ^2 distribution and has k degrees of freedom (df) [29]. The F distribution is the distribution of the quotient of two independent random variables that are each distributed according to a χ^2 distribution and divided by their respective df [29]. This has df k_1 and k_2 where k_1 is the df of the numerator and k_2 is the df of the denominator.

For convenience as well as succinctness of notation, restrictions are sometimes placed on algorithm inputs. The most common of these for process monitoring is assuming that the mean of input signals is 0. To adhere to this restriction, data is preprocessed before being used in the algorithm. For algorithms that take the whole data set as an input, this can be done by determining the means beforehand. For incremental algorithms, the mean can be updated according to Equation (2.4) for variable $x_i^\star$ and step n.

$$\bar{x}_i(n) = \bar{x}_i(n-1) + \frac{x_i^\star(n) - \bar{x}_i(n-1)}{n} \tag{2.4}$$

Another less common restriction is that each signal has a variance of 1. The same preprocessing steps can be performed with the incremental variance calculated using Equation (2.5).

$$s_i^2(n) = \frac{n}{n-1}\left[\frac{\sum_{j=1}^n x_i^\star(j)^2}{n} - \left(\frac{\sum_{j=1}^n x_i^\star(j)}{n}\right)^2\right] \tag{2.5}$$

This requires keeping running tallies of the number of samples, $x_i^\star$, and $(x_i^\star)^2$. The data points are then recalculated as in Equation (2.6).

$$x_i(n) = \frac{x_i^\star(n) - \bar{x}_i}{s_i} \tag{2.6}$$

Some algorithms may whiten the data which means applying a linear transformation such that the transformed data has a covariance matrix equal to the identity matrix of the same size. There are infinitely many ways to whiten data and some of these can be found in [30].

An incremental estimation of the covariance matrix can be found using Equation (2.7).

$$\mathrm{Cov}(\mathbf{X})(n) = \frac{1}{n-1}\sum_{j=1}^{n}\left(\mathbf{x}^{\star}(j) - \bar{\mathbf{x}}(j)\right)\left(\mathbf{x}^{\star}(j) - \bar{\mathbf{x}}(j)\right)^{\top} \tag{2.7}$$

Alternatively, the incremental estimations in Equations (2.4), (2.5), and (2.7) can be updated using a learning rate θ which will skew the estimations to be more representative of recent samples rather than the whole data set.

$$\bar{\mathbf{x}}(n) = (1-\theta)\,\bar{\mathbf{x}}(n-1) + \theta\;\mathbf{x}^{\star}(n) \tag{2.8}$$

$$\mathbf{s}^2(n) = (1-\theta)\,\mathbf{s}^2(n-1) + \theta\left(\mathbf{x}^{\star}(n) - \bar{\mathbf{x}}(n)\right)^2 \tag{2.9}$$

$$\mathrm{Cov}(\mathbf{X})(n) = (1-\theta)\,\mathrm{Cov}(\mathbf{X})(n-1) + \theta\left(\mathbf{x}^{\star}(n) - \bar{\mathbf{x}}(n)\right)\left(\mathbf{x}^{\star}(n) - \bar{\mathbf{x}}(n)\right)^{\top} \tag{2.10}$$

2.2 Slow Feature Analysis

SFA is an unsupervised learning method that transforms a set of m input signals into J slow features where $J \leq m$ [20]. Assuming that the input signals have a mean of 0, a sample is denoted as $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_m(t)]^{\top}$. SFA finds a set of transformations $G(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_J(\mathbf{x})]^{\top}$ that minimizes the speeds of the features $\mathbf{y}(t) = G(\mathbf{x}(t))$. This can be restated as the following optimization problem.

$$\min_{g_j} \;\; \Delta(y_j) \doteq \langle \dot{y}_j^2 \rangle_t = \langle [g_j(\mathbf{x}(t)) - g_j(\mathbf{x}(t-1))]^2 \rangle_t \tag{2.11}$$

Subject to

$$\langle y_j^2 \rangle_t = 1 \tag{2.12}$$

$$\langle y_j y_j' \rangle_t = 0 \qquad \forall\, j' < j \tag{2.13}$$

The angle brackets denotes averaging over time $\langle f \rangle_t = 1/t \int_0^t f(t)dt$ and the dot accent denotes the first derivative $\dot{f} = (d/dt)f(t)$. The variance constraint in Equation (2.12) prevents trivial solutions of $y = c$ where c is some constant. The decorrelation constraint in Equation (2.13) ensures that information is not repeated across features.

Here, input signals refer to outputs from process sensors such as temperature, pressure, and flow rate. Features are variables which are functions of these input signals which contain latent information about the process. Just as pressures might fluctuate over time, so will these features. The slowness of these features is the frequency at which these fluctuations occur.

SFA proposes that the slowest features contain important information about the process while the fastest features contain noise. By extracting the slowest J features, a reduction in dimension is performed with minimal loss of information. A criterion for choosing J was proposed in [24] which only keeps the features that are slower than the fastest q fraction of input signals. The equation for this is

$$J = \text{card}\{y_i | \Delta(y_i) < \max_j^q \{\Delta(x_j)\}\} \tag{2.14}$$

where card is cardinality and counts the number of elements in a set.

For a linear transformation, the features are a weighted combination of the input signals $\mathbf{y} = \mathbf{W}^\top \mathbf{x}$. Matrices $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{Y} \in \mathbb{R}^{J \times n}$ can be used to denote input signals and features for all n samples. Using an additional matrix $\mathbf{D}$ that is equivalent to taking the first backward difference, SFA can be restated as the following optimization problem.

$$\min_{\mathbf{W}} \ \mathbf{Tr}\left(\mathbf{W}^\top \mathbf{B} \mathbf{W}\right) \tag{2.15}$$

Subject to

$$\mathbf{W}^\top \mathbf{A} \mathbf{W} = \mathbf{I} \tag{2.16}$$

where $\mathbf{A} = \text{Cov}(\mathbf{X})$ and $\mathbf{B} = \text{Cov}(\mathbf{X}\mathbf{D})$. This problem can be solved analytically using Algorithm 1 [24] where $\text{SVD}(\mathbf{M})$ is the singular value decomposition (SVD) of a matrix.

Algorithm 1: $\text{SFA}(\mathbf{X}, J)$

Data: $\mathbf{X} \in \mathbb{R}^{m \times n}, J \in \mathbb{N}$
Result: $\mathbf{W} \in \mathbb{R}^{m \times J}, \mathbf{Y} \in \mathbb{R}^{J \times n}$
$\text{Cov}(\mathbf{X}) \leftarrow \frac{1}{n-1} \mathbf{X}\mathbf{X}^\top$;
$\mathbf{U}, \mathbf{S}, \mathbf{U}^\top \leftarrow \text{SVD}(\text{Cov}(\mathbf{X}))$;
$\mathbf{Q} \leftarrow \mathbf{U}\mathbf{S}^{-\frac{1}{2}}$;
$\mathbf{Z} \leftarrow \mathbf{Q}^\top \mathbf{X}$;
$\dot{\mathbf{Z}} \leftarrow \mathbf{Z}\mathbf{D}$; `/* D is the first backward difference matrix */`
$\text{Cov}(\dot{\mathbf{Z}}) \leftarrow \frac{1}{n-1} \dot{\mathbf{Z}}\dot{\mathbf{Z}}^\top$;
$\mathbf{P}, \mathbf{\Omega}, \mathbf{P}^\top \leftarrow \text{SVD}(\text{Cov}(\dot{\mathbf{Z}}))$;
$\mathbf{W} \leftarrow (\mathbf{Q}\mathbf{P})_{[1 \to m, 1 \to J]}$; `/* Only first J columns */`
$\mathbf{Y} \leftarrow \mathbf{W}^\top \mathbf{X}$;

2.3 Incremental Algorithms

It is desirable to have a model that can be updated incrementally given a new sample. One reason for this is that such a model can adapt itself when faced with new situations. This is advantageous for process monitoring as data about every operating condition or fault type is no longer needed; the model can be updated whenever these are first encountered. Incremental updates also mean that older samples can be discarded which reduces the memory and storage requirements which in turn reduces the cost of implementing such methods. These models also remove the need to feed in the whole training set at once which can become intractable for large data sets.

One incremental SFA based algorithm is RSFA which updates the transformation matrix at each iteration. This is done by incrementally updating the covariance matrices. The first SVD operation of SFA is replaced with the rank-one modification algorithm and the second SVD operation is replaced with the orthogonal iteration procedure. This algorithm was developed in [22] with a process monitoring scheme. This algorithm suffers from high complexity which is cubic in terms of the number of input signals. In Chapter 4, an alternative algorithm called IncSFA [21] is adapted for process monitoring. The algorithm was originally applied to high dimensional visual input streams and has a linear complexity in terms of the number of input signals.

2.4 Sparsity

Another desirable model property is sparsity. For techniques such as SFA, sparsity is defined as having a high proportion of 0 values in the transformation matrix. As mentioned previously, the features of SFA are linear combinations of all input signals. Since input signals correspond to physical properties in process monitoring, these models are often difficult to interpret. A sparse model is much easier to interpret and can provide valuable insight into the process. In addition to this, sparsity prevents overfitting of the model to the training data.

Sparsity is measured by the l_0 norm which is a count of the number of non-zero values in a matrix. The inclusion of the l_0 norm leads to NP-hard problems [31] which is why a relaxation to the l_1 norm is often performed. The l_1 norm is a sum of the absolute value of each entry of a matrix. The sparse SFA problem with this relaxation can be stated as

the following optimization problem.

$$\min_{\mathbf{W}} \ \text{Tr}\left(\mathbf{W}^\top \mathbf{B}\mathbf{W}\right) + ||\mathbf{W}||_1 \tag{2.17}$$

Subject to

$$\mathbf{W}^\top \mathbf{A}\mathbf{W} = \mathbf{I} \tag{2.18}$$

2.4.1 Proximal Algorithms

This inclusion of the l_1 norm causes issues by creating a non-smooth objective function. Some specialized algorithms have been created to deal with non-smooth optimization. Some of these are proximal algorithms which are named after the proximal operator that they use. This operator is a convenient shorthand which is defined in Equation (2.19) [32].

$$\mathbf{prox}_{\mu g}(\mathbf{v}) = \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{1}{2\mu}||\mathbf{x} - \mathbf{v}||_2^2 \right) \tag{2.19}$$

In this equation μ is a parameter that controls the trade-off of minimizing the function $g(x)$ and remaining close to the point v. The proximal operator for the l_1 norm is equivalent to the soft thresholding operator $\mathcal{T}_\mu(\mathbf{x})$ which is defined as

$$(\mathcal{T}_\mu(\mathbf{x}))_i = \text{sgn}(x_i)\max(\{|x_i| - \mu, 0\}) \tag{2.20}$$

where $\text{sgn}(\cdot)$ is the sign function [32]. A depiction of this function can be found in Figure 2.1.

When objective functions take the form

$$\min \quad f(\mathbf{x}) + g(\mathbf{x}) \tag{2.21}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are convex and f is differentiable, they can be solved using proximal gradient descent (PGD). This method is outlined in Algorithm 2 [32]. If ∇f is Lipschitz continuous with constant L, this will converge at a rate of $O(1/k)$ when using a fixed step size $\mu \in (0, 1/L]$. An extension of this method that includes an extrapolation step is described in Algorithm 3 which converges at a rate of $O(1/k^2)$ under the same step size constraint [32]. This method is called accelerated proximal gradient descent (APGD).

For sparse problems with a sparsity hyperparameter γ of the form

$$\min_{\mathbf{x}} \ (1/2)||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \gamma||\mathbf{x}||_1 \tag{2.22}$$

PGD and APGD are alternatively named the iterative soft thresholding algorithm (ISTA) and the fast iterative soft thresholding algorithm (FISTA) respectively [32].

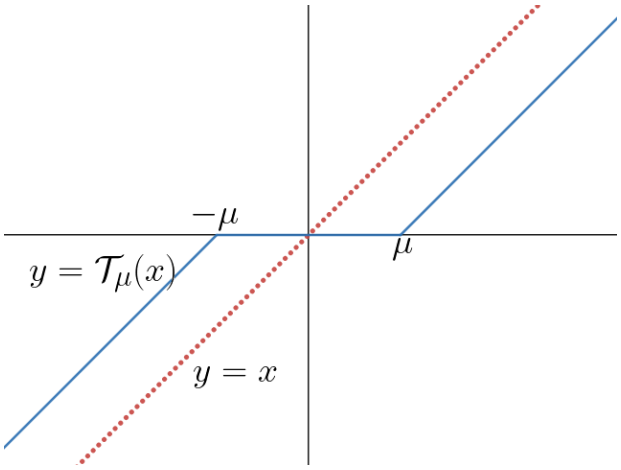

Figure 2.1: Soft thresholding function

Algorithm 2: PGD($\mathbf{x}_0, \mu, \nabla f(\mathbf{x}), \mathbf{prox}_{\mu g}(\mathbf{x})$)

Data: $\mathbf{x}_0, \mu, \nabla f(\mathbf{x}), \mathbf{prox}_{\mu g}(\mathbf{x})$
Result: $\mathbf{x}_{k+1}$
$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_0$;
while *stopping criterion not met* **do**
  $\mathbf{x}_k \leftarrow \mathbf{x}_{k+1}$;
  $\mathbf{x}_{k+1} \leftarrow \mathbf{prox}_{\mu g}(\mathbf{x}_k - \mu \nabla f(\mathbf{x}_k))$;
end

Algorithm 3: APGD($\mathbf{x}_0, \mu, \nabla f(\mathbf{x}), \mathbf{prox}_{\mu g}(\mathbf{x})$)

Data: $\mathbf{x}_0, \mu, \nabla f(\mathbf{x}), \mathbf{prox}_{\mu g}(\mathbf{x})$
Result: $\mathbf{x}_{k+1}$
$\mathbf{x}_k \leftarrow \mathbf{0}$; `/* 0 vector */`
$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_0$;
while *stopping criterion not met* **do**
  $\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k$;
  $\mathbf{x}_k \leftarrow \mathbf{x}_{k+1}$;
  $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_k + \frac{k}{k+3}(\mathbf{x}_k - \mathbf{x}_{k-1})$;
  $\mathbf{x}_{k+1} \leftarrow \mathbf{prox}_{\mu g}(\mathbf{y}_{k+1} - \mu \nabla f(\mathbf{y}_{k+1}))$;
end

2.4.2 Manifold Optimization

According to [33] "a differentiable manifold is a topological space on which there are defined coordinates allowing basic notions of differentiability". Charts are used as bijections from subsets of the manifold set to open subsets of Euclidean space [33]. Using this notion, efficient numerical algorithms can be designed for problems with a manifold structure.

The Stiefel manifold which is the set of all $n \times p$ orthonormal matrices

$$\mathrm{St}(p, n) = \{\mathbf{X} \in \mathbb{R}^{n \times p} | \mathbf{X}^\top \mathbf{X} = \mathbf{I}\} \tag{2.23}$$

is one commonly encountered manifold structure [34]. The generalized Stiefel manifold is a related structure that takes the form of Equation (2.24) [34].

$$\mathrm{St}_{\mathbf{G}}(p, n) = \{\mathbf{X} \in \mathbb{R}^{n \times p} | \mathbf{X}^\top \mathbf{G} \mathbf{X} = \mathbf{I}\} \tag{2.24}$$

This is the same structure as the SFA constraint of Equation (2.16).

An elegant manifold optimization method is the line search algorithm outlined in Algorithm 4 [34]. This requires the use of an operator $R_x(\eta)$ called the retraction which is a smooth mapping of elements from the tangent bundle of a manifold at the point x, $T_x\mathcal{M}$, onto the manifold $\mathcal{M}$ itself. These operators are specific to the manifold being studied and one such operator for the generalized Stiefel manifold is the Cholesky QR-based retraction of [35] outlined in Algorithm 5 where $\mathrm{Chol}(\mathbf{M})$ is the Cholesky factorization of a matrix.

Algorithm 4: ManifoldLineSearch$(t, x_0, T(x), R_x(\eta))$

Data: $t \in \mathbb{R} > 0, x_0 \in \mathcal{M}, T(x) : \mathcal{M} \to T\mathcal{M}, R_x(\eta) : T_x\mathcal{M} \to \mathcal{M}$
Result: $x_{k+1} \in \mathcal{M}$
$x_{k+1} \leftarrow x_0$;
while *stopping criterion not met* **do**
  $x_k \leftarrow x_{k+1}$;
  $\eta_k \leftarrow T(x_k)$;
  $x_{k+1} \leftarrow R_{x_k}(t\eta_k)$;
end

2.4.3 Sparse Slow Feature Analysis

SSFA is an algorithm that is used to solve the problem presented in Equation (2.17) [23]. Instead of optimizing this equation directly, the objective function is transformed

Algorithm 5: CholeskyQRRetraction$(\mathbf{G}, \mathbf{X}, \boldsymbol{\eta})$

Data: $\mathbf{G}, \mathbf{X} \in \mathrm{St}_{\mathbf{G}}, \boldsymbol{\eta} \in T_{\mathbf{X}}\mathrm{St}_{\mathbf{G}}$
Result: $\mathbf{Y} \in \mathrm{St}_{\mathbf{G}}$
$\mathbf{Z} \leftarrow (\mathbf{X} + \boldsymbol{\eta})^{\top}\mathbf{G}(\mathbf{X} + \boldsymbol{\eta})$;
$\mathbf{L}, \mathbf{L}^{\top} \leftarrow \mathrm{Chol}(\mathbf{Z})$;
$\mathbf{Y} \leftarrow (\mathbf{X} + \boldsymbol{\eta})\mathbf{L}^{-1}$;

into a regression formula in the same manner as SPCA [7]. Two matrices are optimized in alternating order before a transformation back into the SFA form is performed. The transformation is computationally complex, and the algorithm is intractable for large data sets. The sparsity of the transformation matrix is also low which means the algorithm is not fully exploiting the advantages of sparse algorithms. Instability also arises for data that has not been normalized. A novel algorithm for the sparse SFA problem that solves these issues is introduced in Chapter 3.

2.5 Anomaly Detection

Unsupervised process monitoring techniques detect anomalous behaviour by detecting when samples exceed some confidence interval on a distribution. If individual variables are assumed to be distributed according to a Gaussian distribution, they can be recentered and rescaled to the standard normal distribution. This allows for the use of the multivariate χ^2 and F distributions to generate confidence intervals. In practice, only the upper limit of the confidence interval (called the control limit or critical value) is needed to detect anomalous behaviour for these 2 distributions since the individual random variables are squared.

Monitoring indices are calculated using sampled data and are then compared against their respective control limit. The Hotelling T^2 and the related S^2 indices were introduced to SFA in [24]. The T^2 index is used to detect static anomalies while the S^2 index is used to detect dynamic anomalies. Other monitoring indices include the squared prediction error (SPE) and combined index which are discussed in [27].

2.6 Fault Diagnosis

It should be noted that while fault detection is another commonly used name for anomaly detection, not all anomalies are faults. For example, if a model was trained on one operating condition only but encountered another normal operating condition during testing, it may (depending on how closely related the two operating conditions are) raise an alarm. A qualified engineer would then judge the alarm and determine the appropriate course of action. This is why fault diagnosis is an important aspect of process monitoring as it will provide the engineer with information as to why the alarm was raised.

A generalized monitoring index is introduced in [27] for fault diagnosis. Using this index, individual signal contributions to the index can be found. Consequently, when a fault has occurred, the contribution values indicate the contribution of each signal to the fault. This information can be used by a qualified engineer to perform a root cause analysis and diagnose the anomalous behaviour.

2.7 Case Studies

2.7.1 Tennessee Eastman Process

The Tennessee Eastman process (TEP) data set is a simulated data set based on a real industrial process [36]. The process uses 5 different input chemicals to create 2 products in 2 separate chemical reactions as well as one byproduct which is produced by 2 different chemical reactions. The main unit operations of this process are a reactor, a stripper, a vapor-liquid separator, a condenser, and a compressor. Since this is a real process, more specific information such as what the different chemicals are has been removed to protect proprietary data. A modified version with a closed-loop plant-wide control scheme is commonly used as a benchmark in process monitoring research [9]. Some fault detection and process monitoring research that uses this data set includes [23, 24, 37, 38, 39]. A diagram of the process with the associated sensors and controllers is shown in Figure 2.2.

The data set includes 22 pairs of training and testing sets which are labelled as IDV(0)-IDV(21) with IDV(0) corresponding to normal operating conditions. The remaining 21 sets simulate different types of faults starting at sample 160 of each set. The types of faults include step changes, random variations, slow drifts, and sticking. The descriptions of these faults are reproduced from [36] in Table 2.1. Since the algorithms used here are unsupervised, they are trained using only the training data for the normal operating condition (IDV(0)).

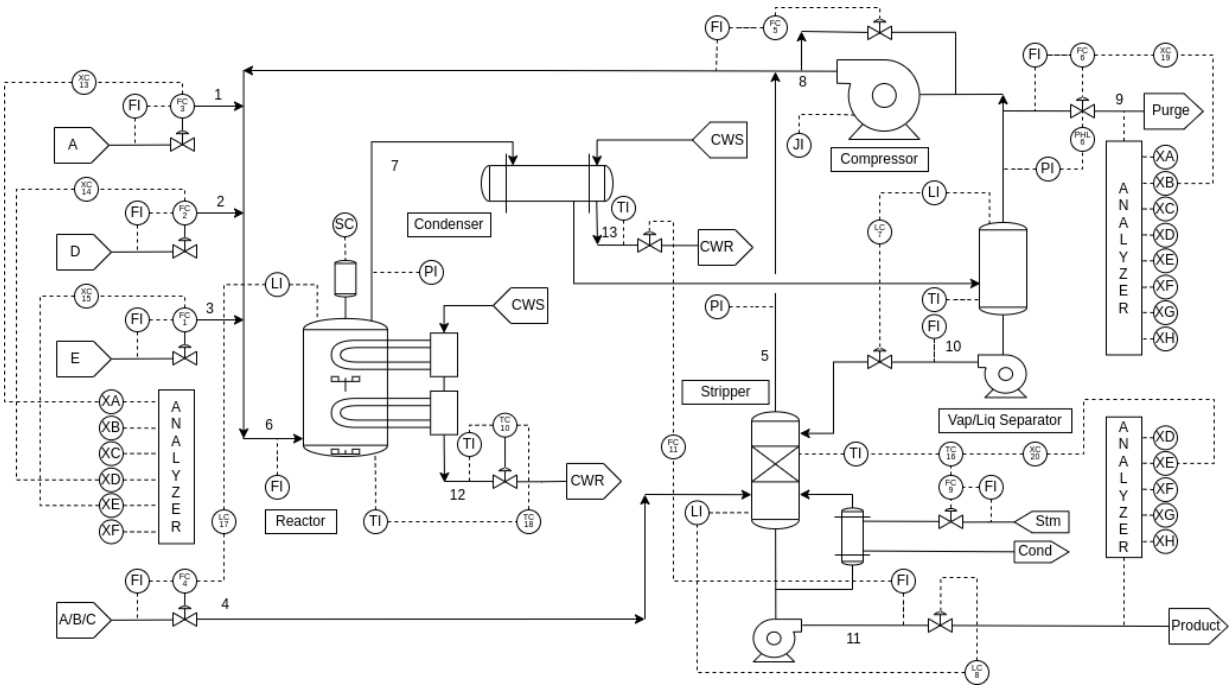

Figure 2.2: Diagram of the Tennessee Eastman Process

The first 22 measured variables and all 11 manipulated variables are used from this data set. These variables are described in Table 2.2. Additionally, 2 lagged samples are appended to the training data as in dynamic PCA [6] to allow time invariant models to capture temporal relationships. The exact number of lags can be chosen with various methods such as auto-correlation analysis [40, 41], Akaike information criterion (AIC) [42, 43], and cross-validation. With these lagged samples, each sample vector is structured as:

$$\mathbf{x}(t) = [x_1(t), \ldots, x_{33}(t), x_1(t-1), \ldots x_{33}(t-1), x_1(t-2) \ldots x_{33}(t-2)]^\top$$

Table 2.1: Fault descriptions of the TEP data set

Fault Label	**Description**	**Fault Type**
IDV(0)	Normal Operation	-
IDV(1)	A/C feed ratio, B composition constant (stream 4)	Step
IDV(2)	B composition, A/C ratio constant (stream 4)	Step
IDV(3)	D Feed Temperature	Step
IDV(4)	Reactor cooling water inlet temperature	Step
IDV(5)	Condenser cooling water inlet temperature	Step
IDV(6)	A feed loss (stream 1)	Step
IDV(7)	C header pressure loss (stream 4)	Step
IDV(8)	A, B, C feed composition (stream 4)	Random Variations
IDV(9)	D feed temperature (stream 2)	Random Variations
IDV(10)	C feed temperature (stream 4)	Random Variations
IDV(11)	Reactor cooling water inlet temperature	Random Variations
IDV(12)	Condenser cooling water inlet temperature	Random Variations
IDV(13)	Reaction kinetics	Slow Drift
IDV(14)	Reactor cooling water valve	Sticking
IDV(15)	Condenser cooling water valve	Sticking
IDV(16)	Unkown	Unkown
IDV(17)	Unkown	Unkown
IDV(18)	Unkown	Unkown
IDV(19)	Unkown	Unkown
IDV(20)	Unkown	Unkown
IDV(21)	Unkown	Unkown

Table 2.2: TEP variable descriptions

Variable	**Description**	**Units**
XMEAS(1)	A Feed (stream 1)	$kscmh$
XMEAS(2)	D Feed (stream 2)	kg/hr
XMEAS(3)	E Feed (stream 3)	kg/hr
XMEAS(4)	A and C Feed (stream 4)	$kscmh$
XMEAS(5)	Recycle Flow (stream 8)	$kscmh$
XMEAS(6)	Reactor Feed Rate (stream 6)	$kscmh$
XMEAS(7)	Reactor Pressure	kPa_{gauge}
XMEAS(8)	Reactor Level	%
XMEAS(9)	Reactor Temperature	$°C$
XMEAS(10)	Purge Rate (stream 9)	$kscmh$
XMEAS(11)	Product Sep Temp	$°C$
XMEAS(12)	Product Sep Level	%
XMEAS(13)	Prod Sep Pressure	kPa_{gauge}
XMEAS(14)	Prod Sep Underflow (stream 10)	m^3/hr
XMEAS(15)	Stripper Level	%
XMEAS(16)	Stripper Pressure	kPa_{gauge}
XMEAS(17)	Stripper Underflow (stream 11)	m^3/hr
XMEAS(18)	Stripper Temperature	$°C$
XMEAS(19)	Stripper Steam Flow	kg/hr
XMEAS(20)	Compressor Work	kW
XMEAS(21)	Reactor Cooling Water Outlet Temp	$°C$
XMEAS(22)	Separator Cooling Water Outlet Temp	$°C$
XMV(1)	D Feed Flow (stream 2)	-
XMV(2)	E Feed Flow (stream 3)	-
XMV(3)	A Feed Flow (stream 1)	-
XMV(4)	A and C Feed Flow (stream 4)	-
XMV(5)	Compressor Recycle Valve	-
XMV(6)	Purge Valve (stream 9)	-
XMV(7)	Separator Pot Liquid Flow (stream 10)	-
XMV(8)	Stripper Liquid Product Flow (stream 11)	-
XMV(9)	Stripper Steam Valve	-
XMV(10)	Reactor Cooling Water Flow	-
XMV(11)	Condenser Cooling Water Flow	-

2.7.2 Three Phase Flow Facility

The three-phase flow facility (TPFF) data set [41] is another benchmark data set. This facility is meant to provide air, water, and oil into a pressurized system. Unlike the TEP set, there are multiple normal operating conditions. The system switches between these operating conditions during the training data. Depending on the test set, a single operating condition is used, or the condition is varied. In total, there are 20 different operating conditions (combinations of the set-points in Table 2.3). The descriptions of the fault sets are described in Table 2.4. The 24 process variables are presented in Table 2.5. The information in these tables has been reproduced from [41].

Since the 2-inch riser is only used for the 6th fault case, both the pressure measurement at PT417 and fault case 6 were excluded for this case study. Fault case 2 was also excluded in the discussion since every method presented in [41] and in this work failed to produce adequate monitoring results. The number of lags chosen for this data set is 2 resulting in a total of 69 input signals.

Table 2.3: TPFF operating set points

Air Flowrate (m^3/s)	**Water Flowrate** (kg/s)
0.0208	0.5
0.0278	1.0
0.0347	2.0
0.0417	3.5
-	6.0

Table 2.4: TPFF fault set descriptions

Data set	**Air Flowrate** (m^3/s)	**Water Flowrate** (kg/s)	**Fault Start** (s)	**Fault End** (s)
Fault 1: Air line blockage (Incipient)				
1.1	Varying	Varying	1566	5181
1.2	0.0417	2.0	657	3777
1.3	0.0208	3.5	691	3691
Fault 2: Water line blockage (Incipient)				
2.1	Varying	Varying	2244	6616
2.2	0.0278	2.0	476	2656
2.3	0.0417	3.5	331	2467
Fault 3: Top separator input blockage (Incipient)				
3.1	Varying	Varying	1136	8352
3.2	0.0278	2.0	333	5871
3.3	0.0208	3.5	596	9566
Fault 4: Open direct bypass (Incipient)				
4.1	Varying	Varying	953	6294
4.2	0.0417	2.0	851	3851
4.3	0.0208	3.5	241	3241
Fault 5: Slugging Conditions (Intermittent)				
5.1	Varying	Varying	686/1772	1172/2253
5.2	Varying	Varying	1633/7031/8057	2955/7553/10608
Fault 6: Pressurization of 2" line (Abrupt)				
6.1	Varying	Varying	1723	2800
6.2	Varying	Varying	1037	4830

Table 2.5: TPFF variable descriptions

Variable	**Location**	**Description**	**Units**
1	PT312	Air delivery pressure	MPa
2	PT401	Pressure in the bottom of the riser	MPa
3	PT408	Pressure in top of the riser	MPa
4	PT403	Pressure in top separator	MPa
5	PT501	Pressure in 3 phase separator	MPa
6	PT408	Diff. pressure (PT401-PT408)	MPa
7	PT403	Differential pressure over VC404	MPa
8	FT305	Flow rate input air	Sm^3/s
9	FT104	Flow rate input water	kg/s
10	FT407	Flow rate top riser	kg/s
11	LI405	Level top separator	m
12	FT406	Flow rate top separator output	kg/s
13	FT407	Density top riser	kg/m^3
14	FT406	Density top separator output	kg/m^3
15	FT104	Density water input	kg/m^3
16	FT407	Temperature top riser	$^{\circ}C$
17	FT406	Temperature top separator output	$^{\circ}C$
18	FT104	Temperature water input	$^{\circ}C$
19	LI504	Level gas-liquid 3 phase separator	%
20	VC501	Position of valve VC501	%
21	VC302	Position of valve VC302	%
22	VC101	Position of valve VC101	%
23	PO1	Water pump current	A
24	PT417	Pressure in mixture zone 2” line	MPa

Chapter 3

Manifold Sparse Slow Feature Analysis

3.1 The Proposed Algorithm

To address the issues in SSFA that were discussed in Section 2.4.3, a new algorithm was created called manifold sparse slow feature analysis (MSSFA). The sparse SFA objective (Equation (2.17)) function can be rewritten as

$$\min_{\mathbf{W}} \quad f(\mathbf{W}) + g(\mathbf{W}) \tag{3.1}$$

where

$$f(\mathbf{W}) = \text{Tr}\left(\mathbf{W}^\top \mathbf{B} \mathbf{W}\right) \tag{3.2}$$

$$g(\mathbf{W}) = ||\mathbf{W}||_1 \tag{3.3}$$

In this formulation, the function $f(\mathbf{W})$ is differentiable and convex while $g(\mathbf{W})$ is non-differentiable and convex. This means that APGD can be used to optimize the function.

The problem with using accelerated proximal gradient descent (APGD) directly is that the algorithm will most likely produce a matrix that does not meet the sparse SFA constraint (Equation (2.18)). The constraint can be rewritten as

$$\mathbf{W} \in \text{St}_{\mathbf{A}}(J, m) \tag{3.4}$$

where

$$\text{St}_{\mathbf{A}}(J, m) = \{\mathbf{X} \in \mathbb{R}^{m \times J} | \mathbf{X}^\top \mathbf{A} \mathbf{X} = \mathbf{I}\} \tag{3.5}$$

The Cholesky QR-based retraction can then be used to meet the constraint by placing the solution on the manifold $\mathrm{St}_{\mathbf{A}}(J, m)$. MSSFA iterates between optimizing the objective function and meeting the constraint until convergence.

The first step is to perform an extrapolation using the previous solution as in APGD

$$\mathbf{U}_k = \mathbf{W}_{k-1} + \frac{k}{k+3}\left(\mathbf{W}_{k-1} - \mathbf{W}_{k-2}\right) \tag{3.6}$$

where k is the number of iterations and $\mathbf{W}_0$ is the first J columns of an identity matrix of size m.

To meet the constraint of Equation (2.18), a retraction is performed

$$\mathbf{V}_k = R_{\mathbf{U_k}}(\alpha_k \boldsymbol{\eta}_k) \tag{3.7}$$

where the step size is chosen as

$$\alpha_k = \frac{1}{k+3} \tag{3.8}$$

to diminish over time. The direction $\boldsymbol{\eta}_k$ is the gradient of $f(\mathbf{W})$ from the objective function divided by its Lipschitz constant in the same manner that the direction of APGD is chosen. The derivation of the Lipschitz constant can be found in Appendix A.

$$\boldsymbol{\eta}_k = -\frac{1}{L}\nabla\left(\mathrm{Tr}\left(\mathbf{W}^\top \mathbf{B}\mathbf{W}\right)\right) = -\frac{\mathbf{BW}}{||\mathbf{B}||_F} \tag{3.9}$$

The notation $||\cdot||_F$ denotes the Frobenius norm.

The proximal operator of the l_1-norm is then applied to $\mathbf{V}_k$ using the reciprocal of the Lipschitz constant as the threshold value to obtain a sparse solution.

$$\mathbf{W}_k = \mathrm{prox}_{L^{-1}||\cdot||_1}\left(\mathbf{V}_k\right) = \mathcal{T}_{(2||\mathbf{B}||_F)^{-1}}\left(\mathbf{V}_k\right) \tag{3.10}$$

Equations (3.6) - (3.10) are repeated until the transformation matrix converges. Pseudocode for the algorithm can be found in Algorithm 6.

3.2 Alternative Regularization

Other regularization methods exist which can also be used to prevent overfitting. The regularizing function $g(\mathbf{W})$ can be modified to any regularization with an efficient or closed

Algorithm 6: MSSFA($\mathbf{W}_0, \mathbf{X}$)

Data: $\mathbf{W}_0 \in \mathbb{R}^{J\times m}, \mathbf{X} \in \mathbb{R}^{m\times n}$
Result: $\mathbf{W}_k \in \mathbb{R}^{m\times J}$
$\mathbf{A} \leftarrow \mathrm{Cov}(\mathbf{X})$;
$\mathbf{B} \leftarrow \mathrm{Cov}(\mathbf{XD})$; `/* D is the first backward difference matrix */`
$L \leftarrow 2||\mathbf{B}||_F$;
$\mathbf{W}_{k-1} \leftarrow \mathbf{0}^{J\times m}$; `/* Zero matrix */`
$\mathbf{W}_k \leftarrow \mathbf{W}_0$;
while *stopping criterion not met* **do**
  $\mathbf{W}_{k-2} \leftarrow \mathbf{W}_{k-1}$;
  $\mathbf{W}_{k-1} \leftarrow \mathbf{W}_k$;
  $\mathbf{V}_k \leftarrow \mathbf{W}_{k-1} + \frac{k}{k+3}(\mathbf{W}_{k-1} - \mathbf{W}_{k-2})$;
  $\alpha_k \leftarrow \frac{1}{k+3}$;
  $\boldsymbol{\eta}_k \leftarrow -\frac{2}{L}\mathbf{BW}$;
  $\mathbf{U}_k \leftarrow \mathrm{CholeskyQRRetraction}(\mathbf{A}, \mathbf{V}_k, \alpha_k\boldsymbol{\eta}_k)$;
  `/* sgn, max, and` $|\cdot|$ `are element-wise operations */`;
  $\mathbf{W}_k \leftarrow \mathrm{sgn}(\mathbf{U}_k)\max\{|\mathbf{U}_k| - \frac{1}{L}\mathbf{I}^{J\times m}, \mathbf{0}^{J\times m}\}$; `/* I is the identity matrix */`
end

form solution to its proximal operator. One common regularization in machine learning is l_2 regularization

$$g(\mathbf{W}) = \frac{1}{2}||\mathbf{W}||_2^2 \tag{3.11}$$

whose proximal operator is the shrinkage operator [32].

$$\mathbf{prox}_{\mu g}(\mathbf{v}) = \left(\frac{1}{1+\mu}\right)\mathbf{v} \tag{3.12}$$

The advantage of this regularization is that the function is strictly convex and has a unique minimum [31]. However, it does not promote sparsity [31].

Another regularization is the elastic net which combines both l_1 and l_2 regularization methods.

$$g(\mathbf{W}) = ||\mathbf{W}||_1 + \frac{\gamma}{2}||\mathbf{W}||_2^2 \tag{3.13}$$

This regularization combines the proximal operators of the l_1 and l_2 norms [32].

$$\mathbf{prox}_{\mu g}(\mathbf{v}) = \left(\frac{1}{1+\mu\gamma}\right)\mathbf{prox}_{\mu||\cdot||_1}(\mathbf{v}) \tag{3.14}$$

Elastic net is strictly convex, and it promotes sparsity. The disadvantage here is that an additional hyperparameter γ must be chosen.

3.3 Fault Detection & Diagnosis Framework

The first step in the fault detection and diagnosis framework is to gather data during the normal operation of a process. Here, normal operation means that the process is producing good quality product and is not showing any anomalous behaviour. For real world processes, choosing this data requires knowledge of the specific process and its expected behaviour. This data is used in Algorithm 6 to find a transformation matrix $\mathbf{W}$. This can be used to generate the feature matrix $\mathbf{Y} = \mathbf{W}^\top\mathbf{X}$. The matrices $\mathbf{X}$ and $\mathbf{Y}$, along with the hyperparameters q and α can be used to calculate J as in Equation (2.14). After the MSSFA model is obtained, the corresponding monitoring framework can be developed to detect anomalies in the test data.

First, the monitoring index T^2 is calculated using the output features

$$T^2 = \mathbf{y}(t)^\top\mathbf{y}(t) \tag{3.15}$$

and its control limit is based on the F distribution [9]

$$\tau_{T^2} = \frac{J(n-1)(n+1)}{n(n-J)} F_\alpha(J, n-J) \tag{3.16}$$

where α is the confidence level. Exceeding this limit indicates that the model has deviated from its normal steady state behaviour. For the works presented in this thesis, α is set to 0.99.

The first backward difference of the features $\dot{\mathbf{Y}} = \mathbf{YD}$ are then calculated. The speeds of the features can then be calculated as

$$\omega_i = \frac{1}{n-1} \sum_{j}^{n-1} (\dot{\mathbf{Y}}_{i,j})^2 \tag{3.17}$$

for each feature i. These speeds are then used to create the matrix $\boldsymbol{\Omega}$ whose entries are $\boldsymbol{\Omega}_{i,i} = \omega_i$ and 0 everywhere else. These can be used to calculate the other monitoring index S^2.

$$S^2 = \dot{\mathbf{y}}(t)^\top \boldsymbol{\Omega}^{-1} \dot{\mathbf{y}}(t) \tag{3.18}$$

Its control limit is also based on the F distribution [24].

$$\delta_{S^2} = \frac{J(n-2)n}{(n-1)(n-J-1)} F_\alpha(J, n-J-1) \tag{3.19}$$

Exceeding this limit indicates that the model has encountered a dynamic anomaly.

These indices can be generalized as

$$\text{Index}_{\mathbf{M}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{M} \mathbf{x} \tag{3.20}$$

with the following matrices for the T^2 and S^2 indices.

$$\mathbf{M}_{T^2} = \mathbf{W}\mathbf{W}^\top \tag{3.21}$$

$$\mathbf{M}_{S^2} = \mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^\top \tag{3.22}$$

This general index can be decomposed into the individual input signal contributions according to the complete decomposition contribution (CDC) [27] method

$$\text{CDC}_i = \mathbf{x}^\top \mathbf{M}^{\frac{1}{2}} \boldsymbol{\xi}_i \boldsymbol{\xi}_i^\top \mathbf{M}^{\frac{1}{2}} \mathbf{x} \tag{3.23}$$

where $\boldsymbol{\xi}_i$ is a vector of the same dimension as $\mathbf{x}$ which is equal to 1 at the ith position and 0 everywhere else. For the S^2 index, $\dot{\mathbf{x}}$ is used rather than $\mathbf{x}$ in Equations (3.20) and (3.23).

During the online stage, new samples are transformed into features using the learned transformation matrix. These features can be used to calculate the T^2 and S^2 statistics according to Equations (3.15) and (3.18). Both values exceeding their control limit is an indication that an abrupt anomaly has been encountered. If only the T^2 value exceeds its control limit, it indicates that there has been a slow drift away from normal operating conditions. If only the S^2 value exceeds its control limit, it indicates that a dynamic anomaly has been encountered. When any anomaly is encountered, an alarm is raised and fault diagnosis using the CDC method is performed. The results of this diagnosis are sent to a qualified engineer who would then decide what step to take next. A flowchart of the MSSFA monitoring and diagnosis framework can be found in Figure 3.1.

3.4 Tennessee Eastman Process Case Study

3.4.1 Sparsity & Interpretability

The sparsity of the various models was tested by extracting the first 55 features of the TEP data set. Additionally, the SPCA algorithm of the Scikit-learn package was used [44, 45] in conjunction with PCA monitoring scheme from [46] for comparison to the SFA based algorithms. 55 principal components were extracted for SPCA. The training data was normalized to ensure that scale had no effect on the weighting of the variables.

Only SPCA and MSSFA showed true sparsity (values exactly equal to 0). Approximate sparsity was measured by counting values below a certain threshold as sparse. This threshold was set to 10^{-12} and the sparsity of a model was calculated as

$$\text{Sparsity Fraction} = \frac{\text{card}\{w_{i,j} \mid |w_{i,j}| \leq 10^{-12}\}}{mJ} \tag{3.24}$$

where $w_{i,j}$ is a value in the transformation matrix $\mathbf{W}^\top$. The sparsity fractions were 0, 0.290, 0.953, and 0.758 for SFA, SSFA, SPCA, and MSSFA respectively.

Visualizations of the transformation matrices can be found in Figure 3.2. In this figure, the white spaces indicate sparse values. The values are well distributed in SSFA and SPCA. SFA seems to favor a few input signals in all of its features. MSSFA focuses on the first half of the input signals. This makes sense considering the 2 lagged samples that were

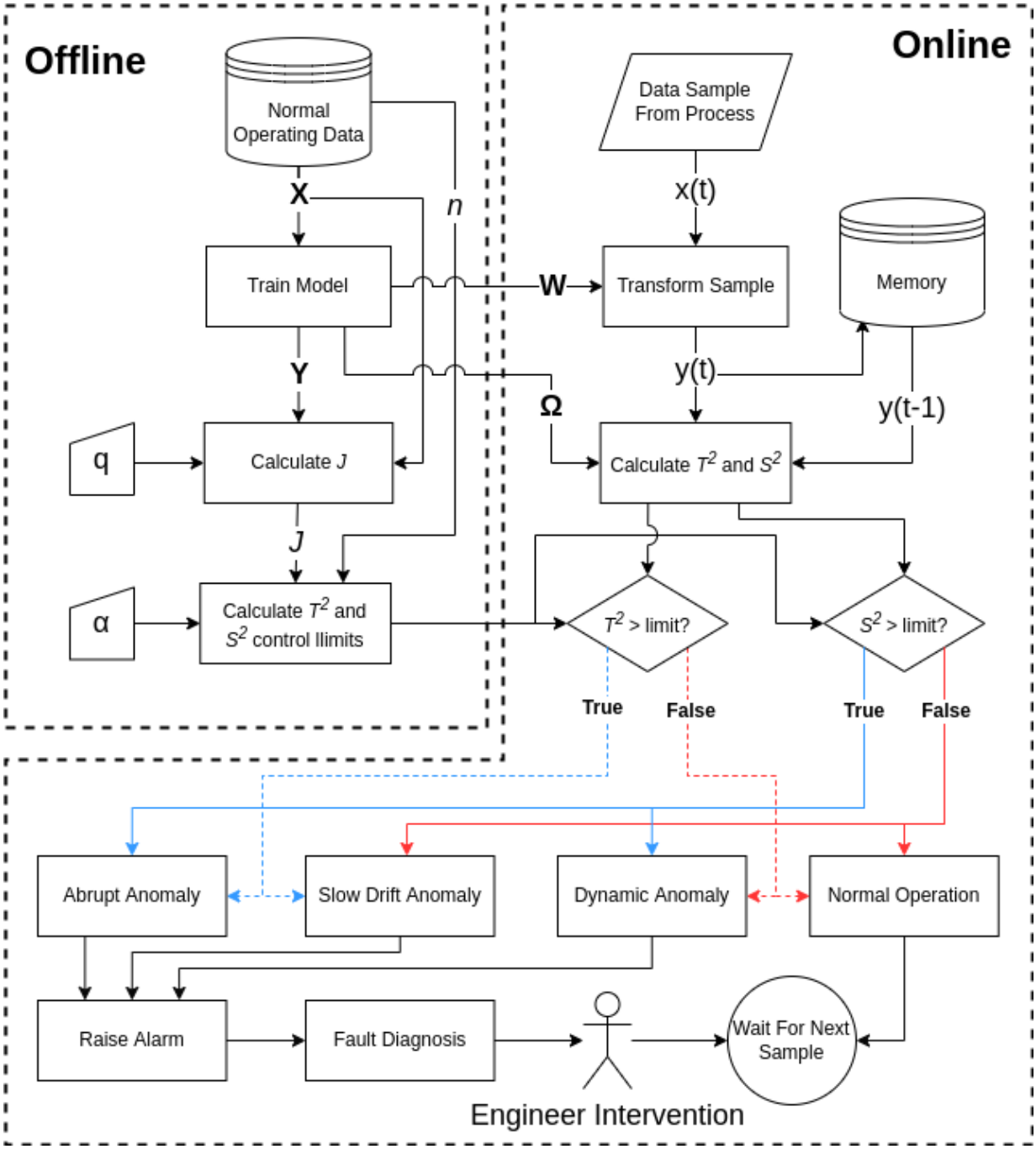

Figure 3.1: MSSFA process monitoring framework

Figure 3.2: Transformation matrices of batch algorithms

appended to the data. Since SFA based algorithms inherently consider process dynamics, the additional slowness from including these signals may not outweigh the increase in sparsity from excluding them.

To get a sense of how interpretable the models are, the makeup of the slowest feature can be examined. For SFA, the largest 10 contributors to the slowest feature accounted for 87.34% of the total contributions. These contributions were the levels and liquid product flows of the stripper and separator. This would indicate that the feature models the effect that manipulating the liquid flow has on the level of the two unit operations. The interpretability suffers however when considering that these variables make significant contributions to most features as seen in Figure 3.2.

The largest 10 contributors to the slowest feature of SSFA accounted for only 61.24% of the total contributions. At first glance, it would seem that this feature models process inputs as the feed flow rates of the reactor feed and inlet streams 1 and 3 are 6 of the 10 largest contributors. The interpretation is less clear when considering that the largest contributor overall with a contribution of 18.07% is the stripper steam valve from the second lagged sample.

The first principal component of SPCA had 90.47% of its contributions from 10 variables. This component most likely models the change in pressure across samples as the pressures of the stripper, reactor, and product separator for the current and lagged samples were 9 of the 10 largest contributors. Unlike the other SFA based algorithms, SPCA must

make use of these lagged samples to model process dynamics.

Only 6 variables accounted for the slowest feature of MSSFA demonstrating the superior interpretability of this model. This feature models the relationship between reactor pressure and process inputs. The 6 variables it uses are the reactor pressure, reactor feed rate, and the inlet streams 1-4. While streams 1-3 combine with the recycle to form the reactor feed, stream 4 is fed into the stripper. Stream 4 is most likely included since its flow is regulated by the reactor level indicator in the control scheme.

The makeup of the features impacts input signal contributions when performing fault diagnosis. They also give meaning to the slow features themselves. For example, in the MSSFA model, a spike in the slowest feature value would indicate anomalous behaviour in the reactor inputs and resulting pressure. Interpretability also provides insight into the process itself by showing the relationships of input signals in the process.

3.4.2 Complexity & Runtime

The computational complexity of an algorithm is an important consideration when working with larger data sets. In this section, the complexity of the algorithms investigated will be given in terms of the number of input signals m, the number of samples n, and the number of desired output features J. For these algorithms $J \leq m$ and in general, it is expected that $m < n$. Additionally, for SSFA and MSSFA, the number of algorithm iterations k and FISTA iterations k' are included.

For SFA, the largest costs incurred come from the creation of the covariance matrix which is $\mathcal{O}(nm^2)$ and SVD which is $\mathcal{O}(m^3)$ leading to a complexity of $\mathcal{O}(nm^2 + m^3)$. The inclusion of sparsity leads to a higher complexity for both SSFA and MSSFA. SSFA complexity arises from the use of SVD, the transformation of data at each iteration $\mathcal{O}(knm^2J)$, and the use of FISTA $\mathcal{O}(kk'm^2J)$ for a final complexity of $\mathcal{O}(m^3 + knm^2J + kk'm^2J)$. MSSFA shows an improved complexity of $\mathcal{O}(nm^2 + km^2J)$ which is due to the calculation of a covariance matrix and the retraction at each iteration $\mathcal{O}(km^2J)$. This beats the cubic complexity of SFA for low values of J. The complexity of SPCA is given in [44] as $\mathcal{O}(km^2 + kJ^2 \max\{n, m\} + kmJ \max\{\sqrt{m}, n\})$.

Each algorithm was applied to the TEP training set to determine their runtimes. The algorithms were asked to extract 10 features from this set as the number of lagged samples was varied resulting in m values ranging from 33-330. The number of samples n ranged from 500-491 due to each additional lagged sample reducing n by 1. A second experiment was performed using a fixed lag number of 2 resulting in 99 input signals. The number

of extracted features J were then varied from 9-99. For both experiments, the algorithms were all run 5 times and the average times of those runs was taken.

The SFA based algorithms were implemented in Python using the NumPy package [47] and the code was run on a 3.60GHz Intel i3-9100F CPU. The runtimes of the two experiments are plotted in Figures 3.3 and 3.4. Note that both graphs use a logarithmic scale for runtime.

The fastest algorithm was SFA for all values across both experiments. It did not change for the varying J values since it always extracts m features. This was consistent with the complexity given above. The runtime of MSSFA had the next fastest runtimes, and it approached the runtime of SFA as the value of m increased. This is consistent with their complexities as SFA is cubic in terms of m while MSSFA is only quadratic. It showed non-linear increase with J as the increase in J demands that more iterations be run for convergence which results in a larger k value. SPCA and SSFA were in competition for the slowest spot as the m values were varied. However, as J increased, the difference in runtimes was more dramatic and SSFA was found to be the slowest. Just like with MSSFA, the nonlinear increase can be attributed to the dependence of k and k' on J. The unintuitive shape of the SPCA trends may be attributable to the random initialization of two matrices used in that algorithm.

3.4.3 Fault Detection

The algorithms were trained using the IDV(0) set of the TEP data. The fault detection rate (FDR) and false alarm rate (FAR) values were found using the T^2 monitoring statistic. FDR is the fraction of faulty samples that were correctly identified and FAR is the fraction of normal samples that were identified as faults. Higher FDR and lower FAR values indicate superior performance.

The criterion mentioned in Equation (2.14) was used to determine the J values of the SFA based algorithms with $q = 0.1$. These were found to be 55, 74, and 85 for SFA, SSFA, and MSSFA respectively. The number of principal components kept in SPCA was chosen as 48 to retain 90% of the variance in the original signals. The results of these algorithms can be found in Table 3.1. In this table, bolded values indicate the best performance on each set but no values are bolded in the case of a tie.

MSSFA had the best FAR for almost all test sets and had the best average FDR and FAR. This is due to its inclusion of dynamics as a SFA based algorithm and its lack of overfitting as a sparse model. SSFA did show a superior FDR in some of the more difficult test sets but the associated increase in FAR for those sets shows that this was most likely

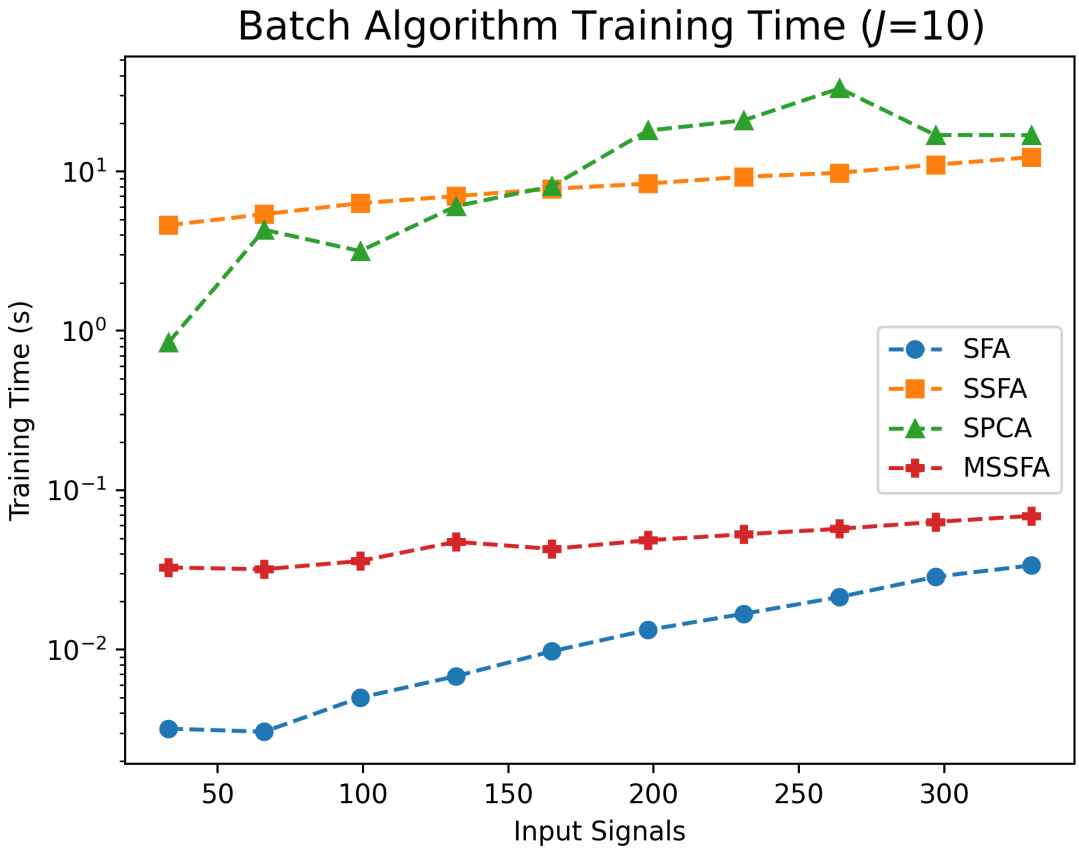

Figure 3.3: Training times of the batch algorithms for a varying number of input signals and constant number of features extracted

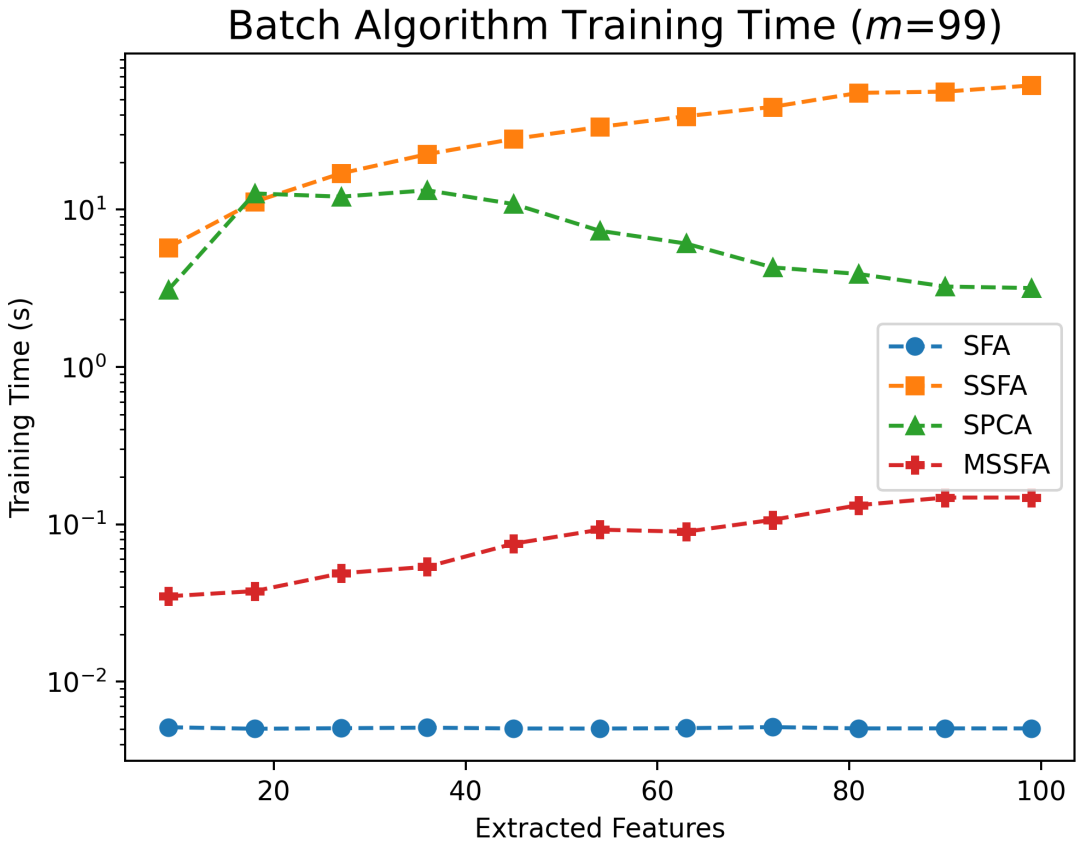

Figure 3.4: Training times of the batch algorithms for a varying number of extracted features and constant number of input signals

Figure 3.5: SFA monitoring statistics for IDV(4)

due to a lack of sensitivity. The dynamic nature of the SFA based algorithms allowed them to outperform SPCA in terms of FDR.

All algorithms performed poorly on IDV(3), IDV(9), and IDV(15). These faults do not cause the process to significantly deviate from its normal operating condition, and so generic monitoring methods can not be used to accurately identify them [24].

As a case study, the monitoring statistics of IDV(4) and IDV(11) were plotted. Both IDV(4) and IDV(11) are changes in the reactor cooling water inlet temperature, but a step change happens in IDV(4) while random variations happen in IDV(11). The IDV(4) plots are Figures 3.5 - 3.8 and the IDV(11) plots are Figures 3.9 - 3.12. In these figures, the solid horizontal black line is the control limit and the vertical dashed line indicates the start of the fault.

All the algorithms detected the introduction of IDV(4). This is characterized both by the step change in T^2 values indicating the move to a new operating condition and the spike in S^2 values indicating an abrupt change in the model. SFA and SPCA however had their T^2 values return to below the limit after the initial change which degraded their performance. The SPE values of SPCA were better able to distinguish the two operating conditions.

Table 3.1: Comparison of FDRs and FARs for the batch algorithms using the TEP dataset

	SFA		SSFA		SPCA		MSSFA	
Test Set	FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR
IDV(0)	0.000	0.037	0.000	0.217	0.000	0.048	0.000	**0.017**
IDV(1)	**1.000**	0.044	0.998	0.032	0.998	0.000	0.999	0.000
IDV(2)	0.970	0.025	0.985	0.000	**0.990**	0.006	0.988	0.000
IDV(3)	0.046	0.057	**0.386**	0.342	0.051	**0.006**	0.044	0.032
IDV(4)	0.514	0.013	1.000	0.070	0.665	0.019	1.000	**0.006**
IDV(5)	1.000	0.013	0.496	0.070	0.294	0.019	1.000	**0.006**
IDV(6)	1.000	0.019	1.000	0.000	0.994	0.006	1.000	0.000
IDV(7)	0.921	0.006	1.000	0.070	1.000	0.032	1.000	0.006
IDV(8)	0.876	0.057	**1.000**	0.272	0.980	**0.006**	0.982	0.013
IDV(9)	0.038	**0.051**	**0.348**	0.437	0.058	0.095	0.029	0.063
IDV(10)	0.858	0.019	0.736	0.057	0.482	0.019	**0.941**	0.019
IDV(11)	0.690	0.025	0.881	0.108	0.755	0.025	**0.899**	**0.000**
IDV(12)	0.998	0.025	**1.000**	0.241	0.991	0.013	0.999	**0.000**
IDV(13)	0.942	0.019	**0.964**	0.019	0.952	**0.000**	0.961	0.006
IDV(14)	1.000	0.038	0.960	0.114	1.000	0.019	1.000	**0.006**
IDV(15)	0.336	0.082	**0.346**	0.044	0.121	0.032	0.204	**0.019**
IDV(16)	0.822	**0.019**	0.758	0.620	0.294	0.152	**0.964**	0.025
IDV(17)	0.956	0.038	0.956	0.203	0.924	0.013	**0.979**	**0.006**
IDV(18)	0.906	0.013	**0.942**	0.025	0.901	0.032	0.908	**0.000**
IDV(19)	0.978	0.013	0.339	0.127	0.434	0.000	**0.995**	0.000
IDV(20)	0.715	0.044	0.764	0.013	0.626	0.000	**0.918**	0.000
IDV(21)	0.239	0.044	**0.606**	0.152	0.564	0.025	0.586	**0.019**
Average	0.753	0.032	0.784	0.147	0.670	0.026	**0.828**	**0.011**

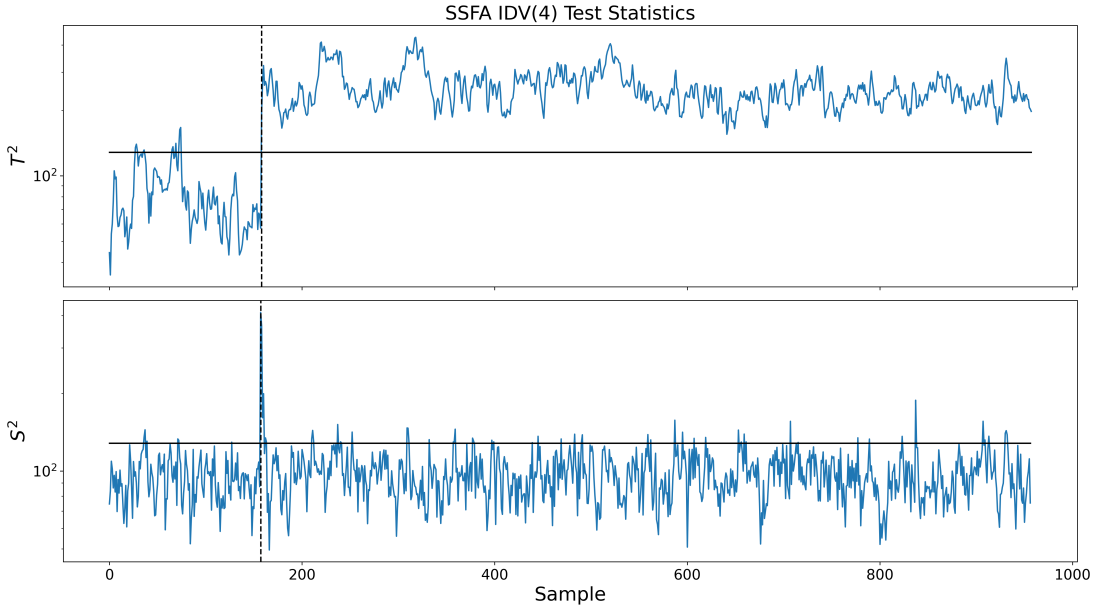

Figure 3.6: SSFA monitoring statistics for IDV(4)

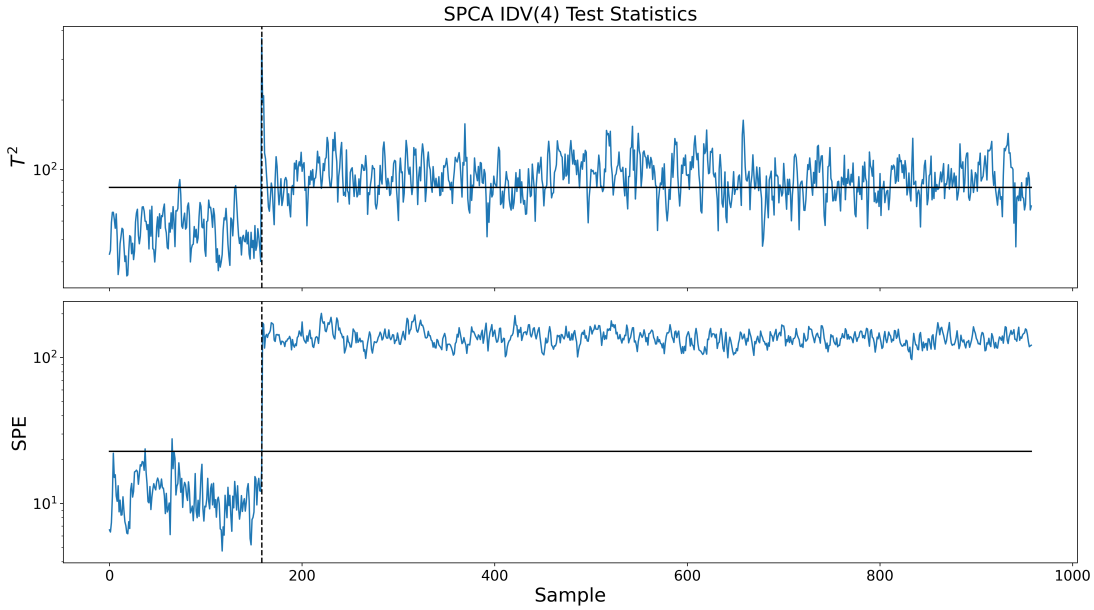

Figure 3.7: SPCA monitoring statistics for IDV(4)

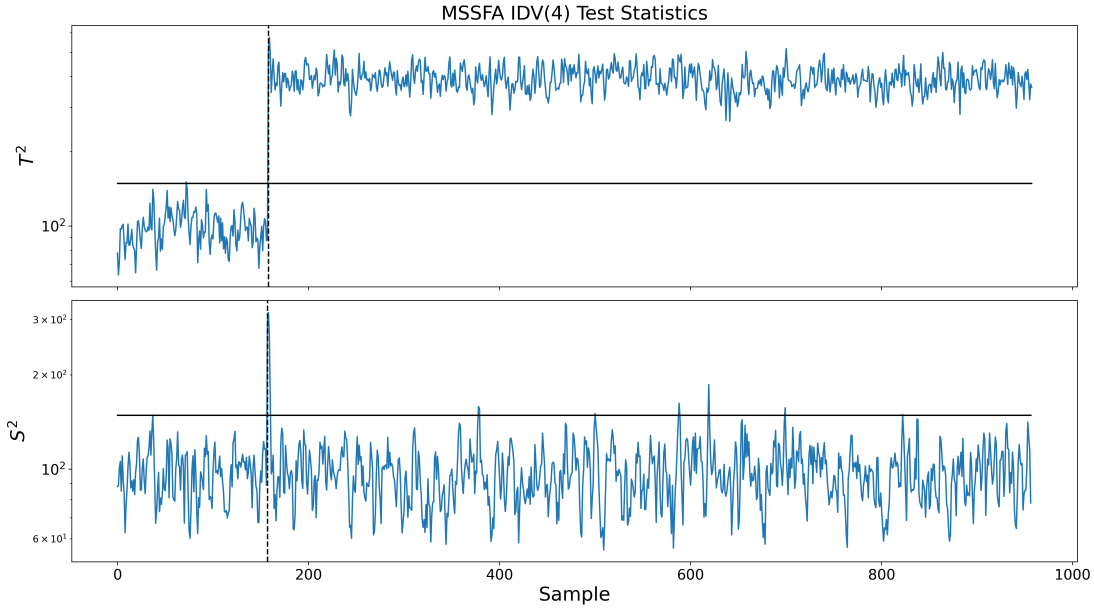

Figure 3.8: MSSFA monitoring statistics for IDV(4)

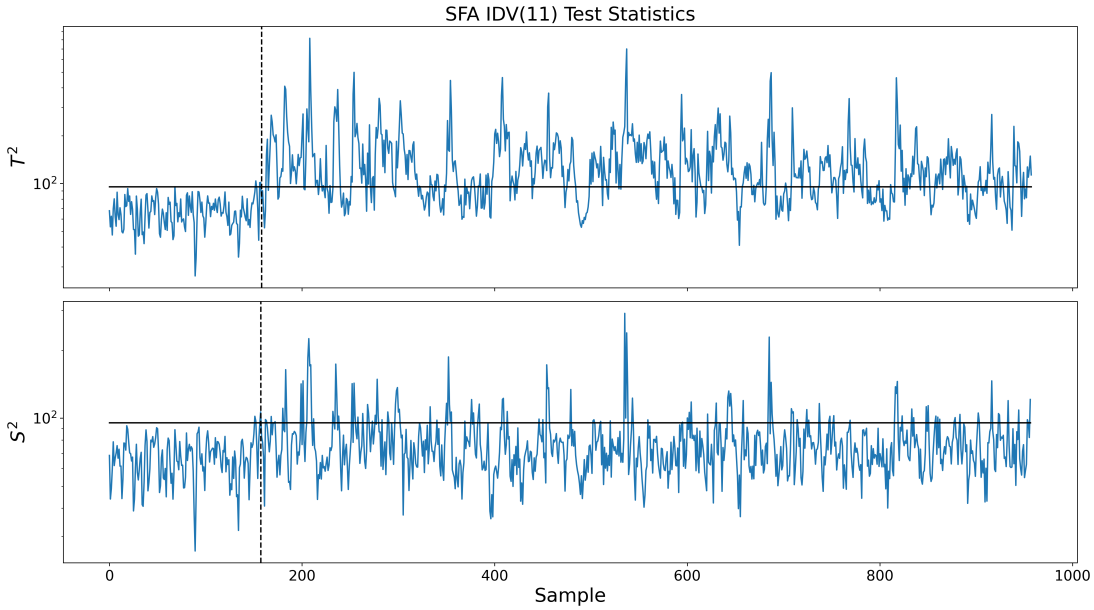

Figure 3.9: SFA monitoring statistics for IDV(11)

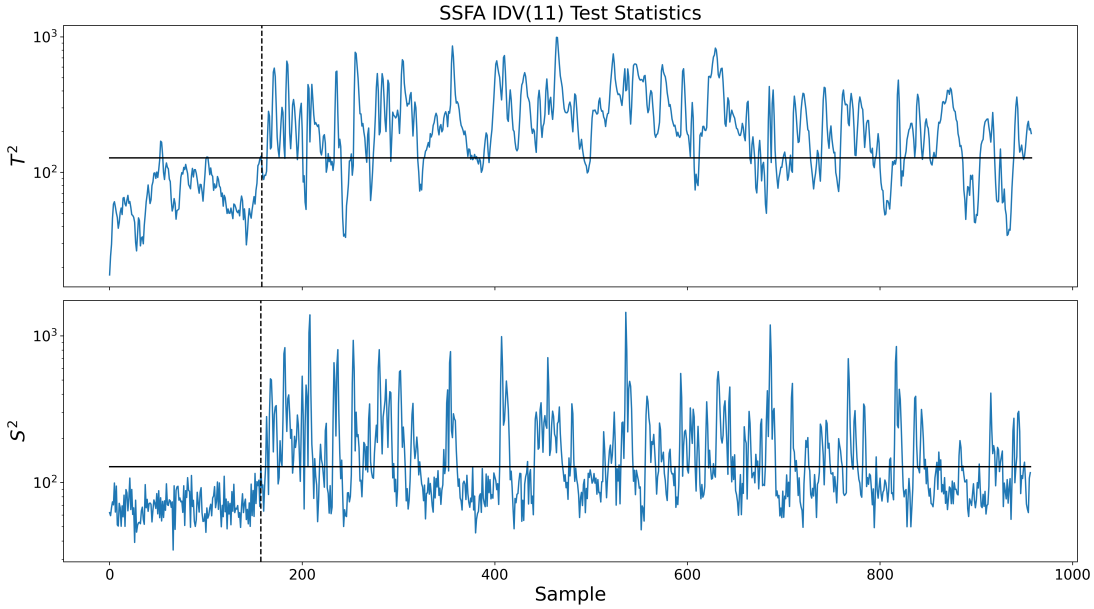

Figure 3.10: SSFA monitoring statistics for IDV(11)

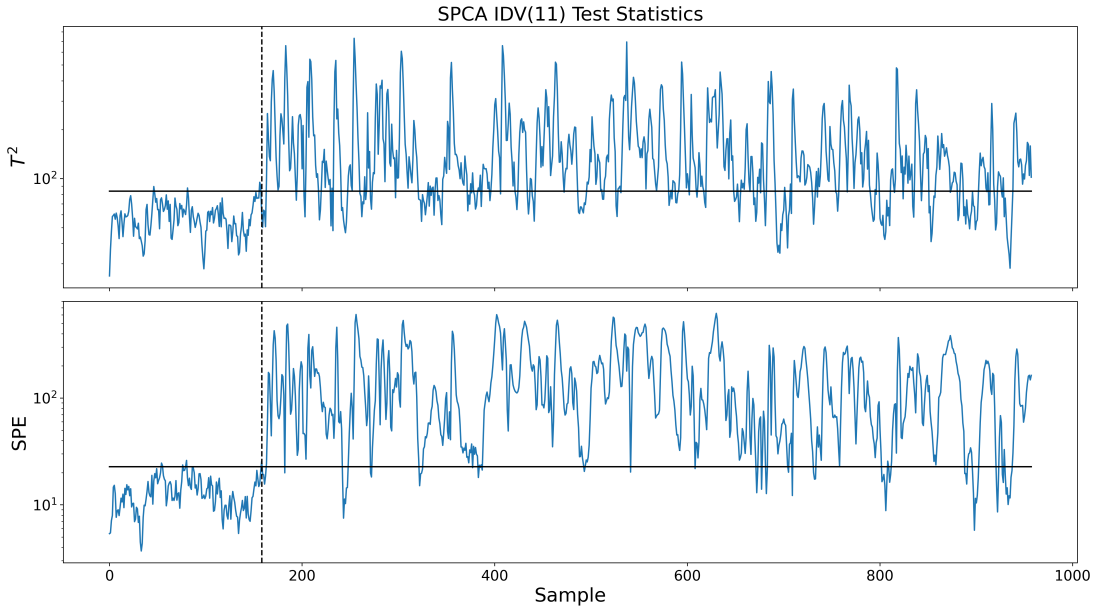

Figure 3.11: SPCA monitoring statistics for IDV(11)

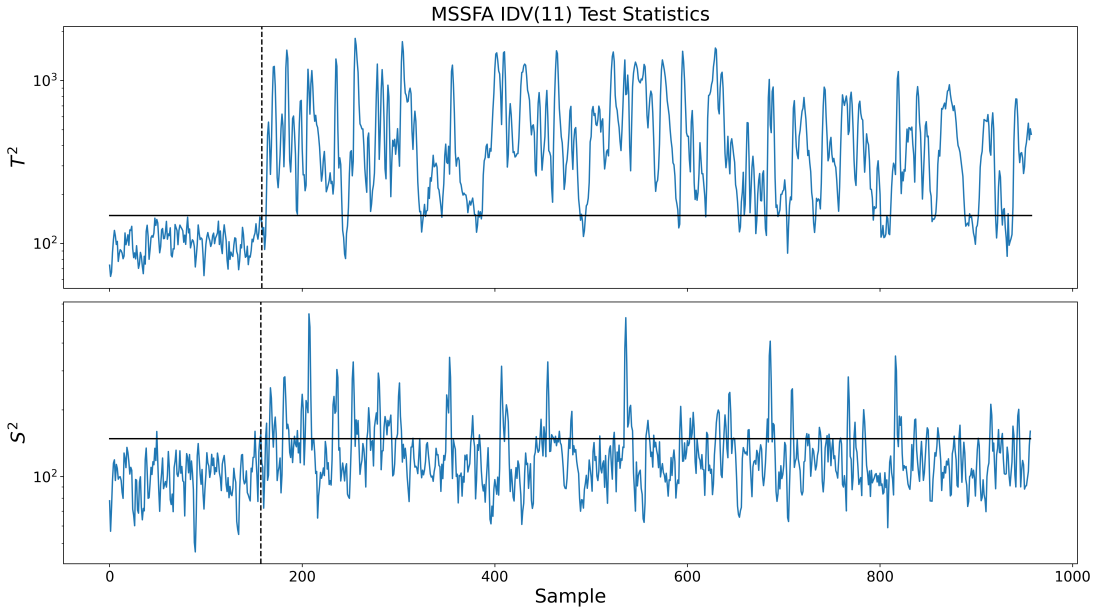

Figure 3.12: MSSFA monitoring statistics for IDV(11)

The performance of the algorithms on IDV(11) was mixed. All algorithms were able to detect at least some spikes corresponding to the random variations, with some detecting more than others. SSFA also raised a few false alarms before the introduction of the fault. MSSFA achieved the highest rate of fault detection without raising any false alarms. For all algorithms, the S^2 values spiked as the operating conditions changed showing lots of abrupt anomalies that would correspond to random variations.

3.4.4 Fault Diagnosis

The case studies of IDV(4) and IDV(11) were used to assess the fault diagnosis performance of the various algorithms. For each algorithm, the CDC contributions of the top 5 contributors were plotted over a certain range. For IDV(4), the range was chosen to coincide with the introduction of the step change.

These plots can be found in Figures 3.13 - 3.16. The algorithms correctly diagnosed the fault and placed the blame of the alarm on the reactor temperature and reactor cooling water flow. SFA also showed some contribution from the unrelated stripper steam flow. SSFA showed contributions from the condenser cooling water flow which is also unrelated

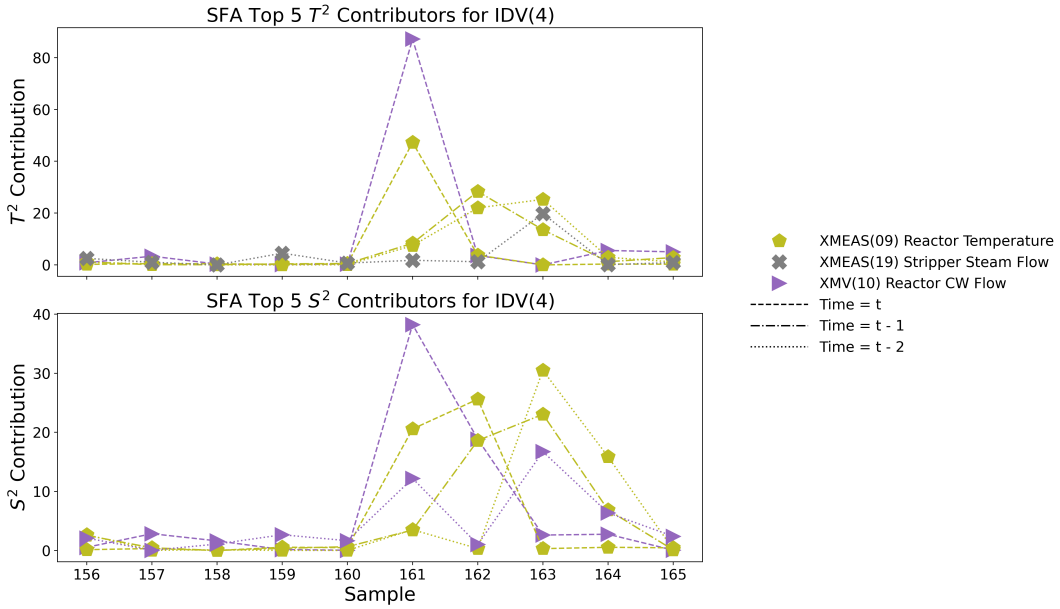

Figure 3.13: SFA fault diagnosis results for IDV(4)

to the fault.

Figures 3.17 - 3.20 show the contributions for IDV(11) at around sample 210. This was considered a long enough period after the introduction of the fault to allow the random variations to affect the whole system. The challenge now becomes to differentiate from these secondary effects and correctly diagnose the fault. SFA now shows additional significant contributions from the product separator pressure. The performance of SSFA diagnosis suffered with the stripper underflow and condenser cooling water flow showing major contributions to the fault. The remainder of the algorithms had a similar diagnosis performance as their IDV(4) counterparts.

3.4.5 Alternative Regularization

The l_2 and elastic net regularization methods were implemented and compared against the l_1 regularization. For elastic net, γ was set to 1. The J values were found to be 61 and 91 for l_2 and elastic net respectively.

The TEP set was used to compare the fault detection performance of the various methods and the results can be found in Table 3.2. l_2 regularization outperforms or matches

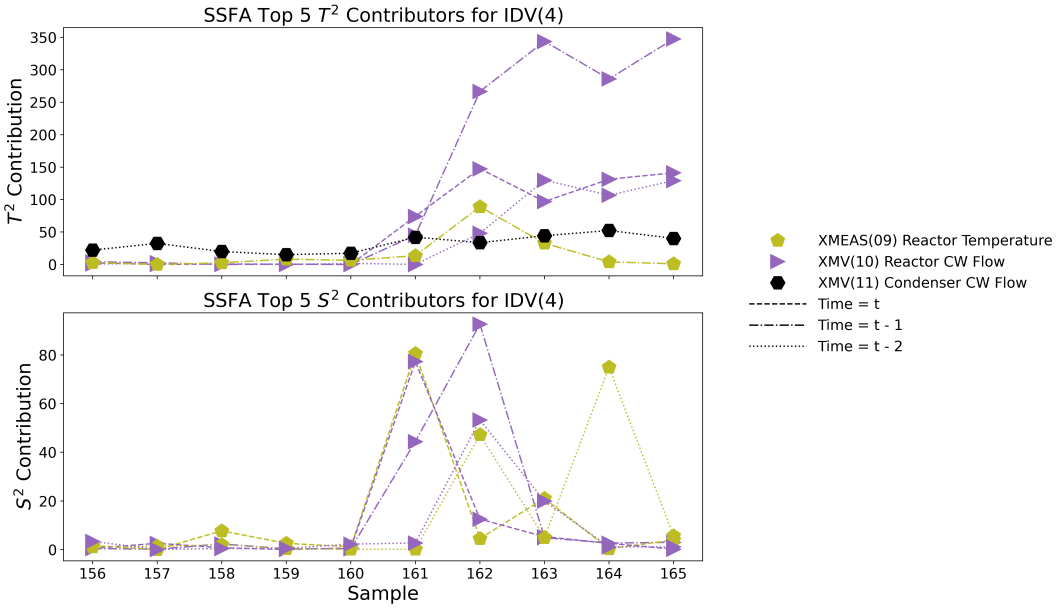

Figure 3.14: SSFA fault diagnosis results for IDV(4)

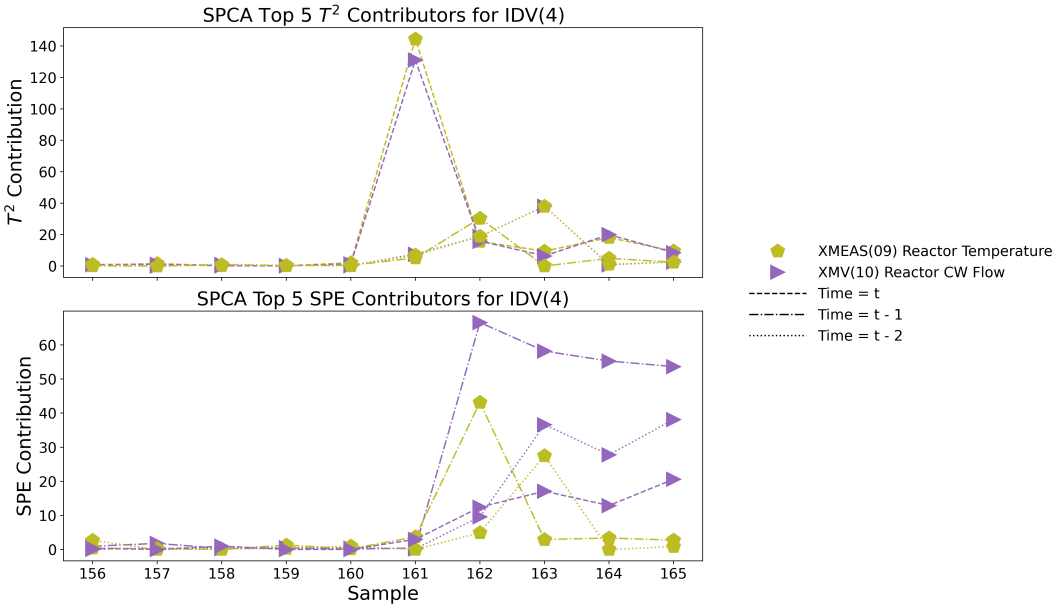

Figure 3.15: SPCA fault diagnosis results for IDV(4)

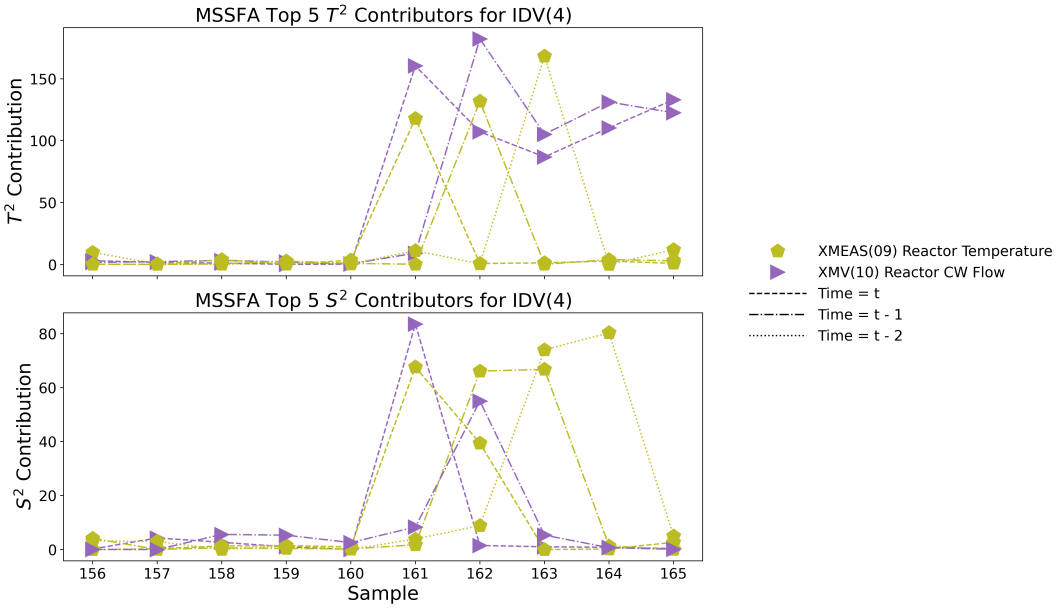

Figure 3.16: MSSFA fault diagnosis results for IDV(4)

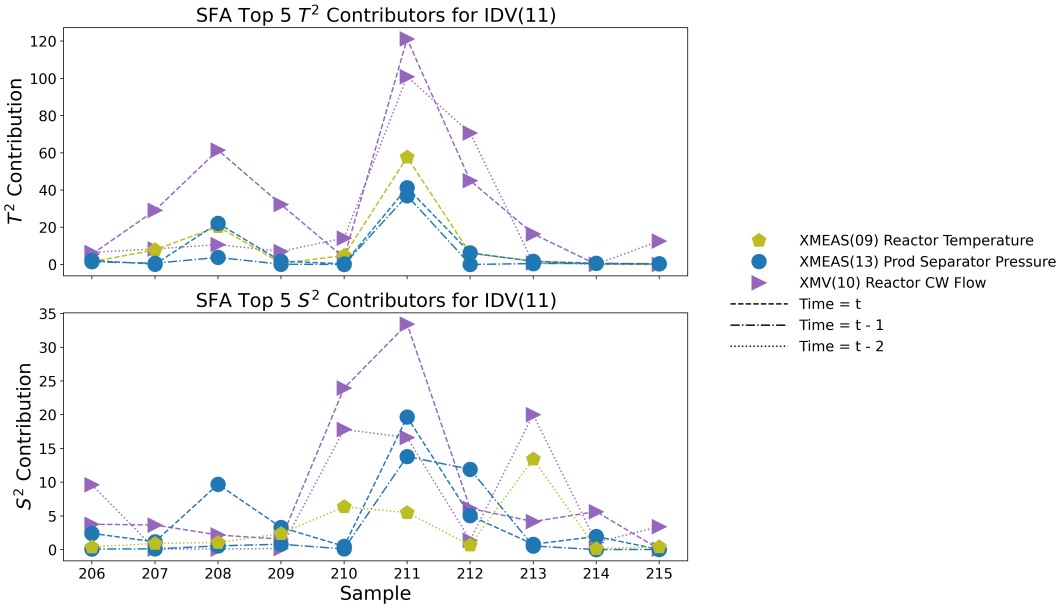

Figure 3.17: SFA fault diagnosis results for IDV(11)

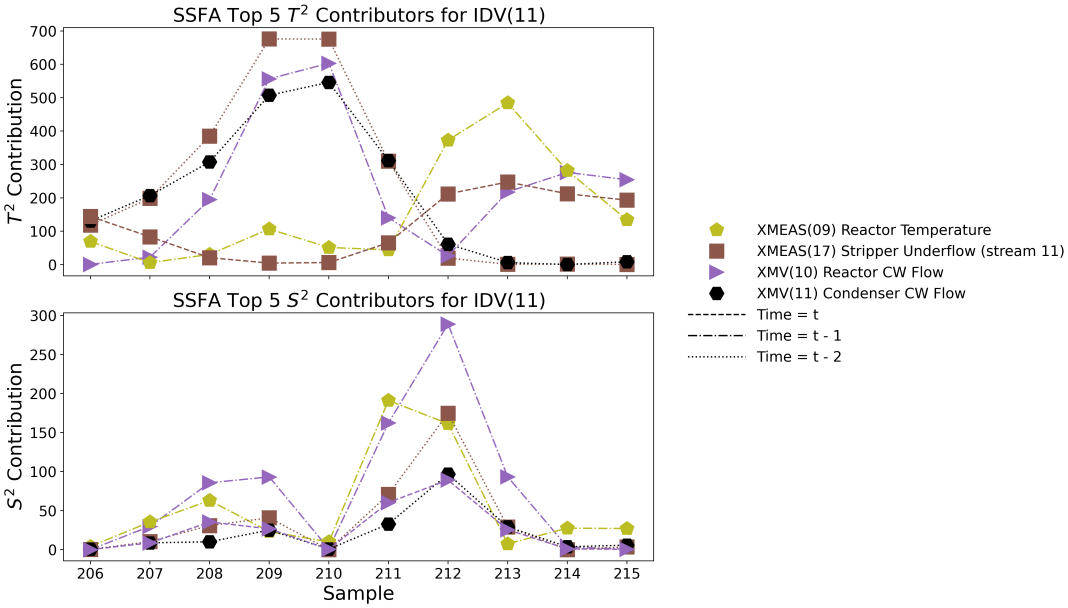

Figure 3.18: SSFA fault diagnosis results for IDV(11)

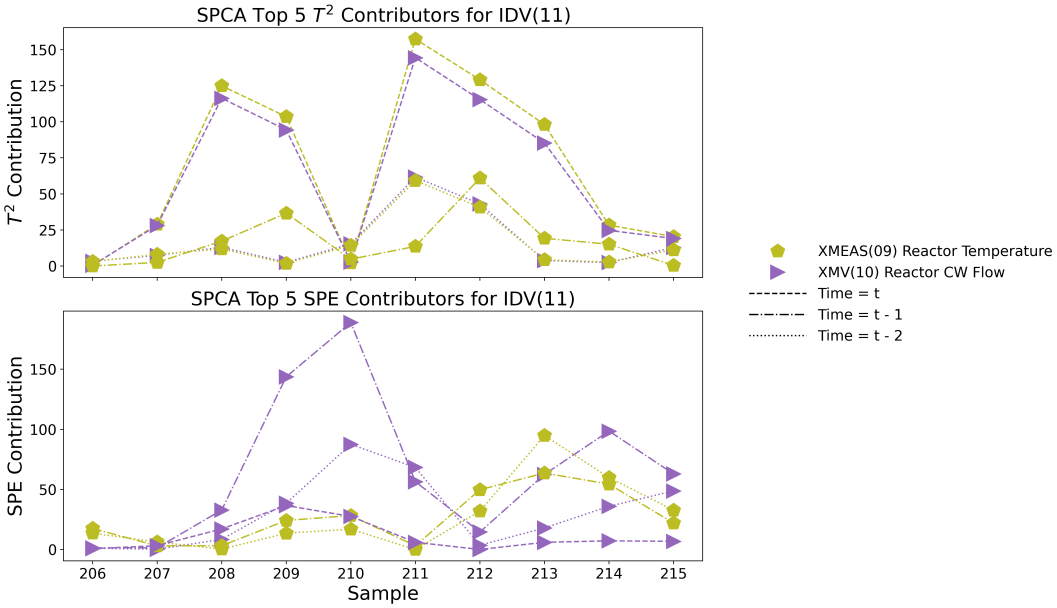

Figure 3.19: SPCA fault diagnosis results for IDV(11)

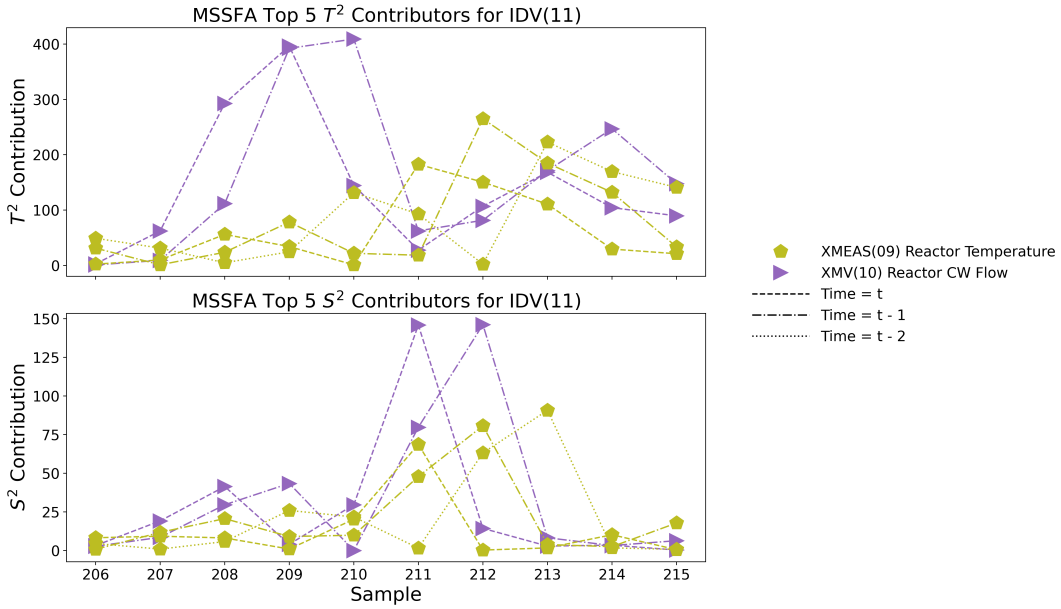

Figure 3.20: MSSFA fault diagnosis results for IDV(11)

the other methods in FDR in each set, but it also has the worst FAR values for each set. The opposite is true with elastic net as it has the best FAR values but the worst FDR values. l_1 values either match the other methods or they lie in between them for both FDR and FAR values. The lower FAR values of l_1 and elastic net provide evidence to the benefits of sparsity.

The sparsity of the algorithms was also compared. Visualizations of the transformation matrices is plotted in Figure 3.21. As expected, the l_2 regularization did not provide a sparse solution. The l_1 and elastic net arrived at similar solutions and had equal sparsity fractions of 0.758.

3.5 Three Phase Flow Facility Case Study

3.5.1 Fault Detection

The models were trained on the TPFF data set using training sets 2 and 3 as recommended by [41]. According to the criterion given in Equation (2.14), a q value of 0.1 meant that

Table 3.2: Comparison of FDRs and FARs for different regularization methods

	l_1		l_2		Elastic Net	
Test Set	FDR	FAR	FDR	FAR	FDR	FAR
IDV(0)	0.000	0.017	0.000	0.087	0.000	**0.009**
IDV(1)	0.999	0.000	**1.000**	0.025	0.999	0.000
IDV(2)	0.988	0.000	**0.994**	0.044	0.986	0.000
IDV(3)	0.044	0.032	**0.174**	0.234	0.018	**0.019**
IDV(4)	1.000	0.006	1.000	0.057	1.000	**0.000**
IDV(5)	1.000	0.006	1.000	0.057	1.000	**0.000**
IDV(6)	1.000	0.000	1.000	0.044	1.000	0.000
IDV(7)	1.000	0.006	1.000	0.019	1.000	**0.000**
IDV(8)	0.982	0.013	**0.989**	0.057	0.981	**0.000**
IDV(9)	0.029	0.063	**0.144**	0.184	0.014	**0.025**
IDV(10)	0.941	0.019	**0.952**	0.057	0.931	**0.000**
IDV(11)	0.899	0.000	**0.950**	0.057	0.880	0.000
IDV(12)	0.999	0.000	0.999	0.038	0.999	0.000
IDV(13)	0.961	0.006	**0.968**	0.032	0.960	**0.000**
IDV(14)	1.000	0.006	1.000	0.044	1.000	**0.000**
IDV(15)	0.204	0.019	**0.371**	0.076	0.144	**0.006**
IDV(16)	0.964	0.025	**0.965**	0.196	0.954	**0.013**
IDV(17)	0.979	0.006	**0.981**	0.082	0.978	0.006
IDV(18)	0.908	0.000	**0.920**	0.038	0.909	0.000
IDV(19)	0.995	0.000	**1.000**	0.025	0.991	0.000
IDV(20)	0.918	0.000	**0.921**	0.025	0.916	0.000
IDV(21)	0.586	0.019	**0.692**	0.114	0.549	**0.013**
Average	0.828	0.011	**0.858**	0.072	0.819	**0.005**

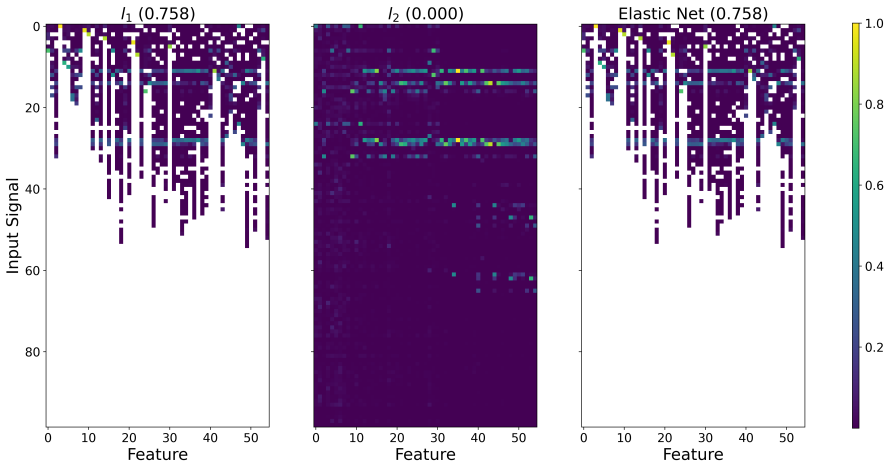

Figure 3.21: Transformation matrices of MSSFA using different regularization methods

17, 61, and 14 features were extracted for SFA, SSFA, and MSSFA respectively. SPCA extracted 8 principal components to retain 90% of the variance. The detection results of the algorithms are given in Table 3.3.

For this data set, SFA showed the lowest FAR values across almost all data sets and its average FDR was better than both SSFA and SPCA. MSSFA had the best FDR performance for almost all the data sets but its average FAR was only better than SSFA and comparable to SPCA. The poor combined performance of SPCA seems to indicate that the dynamics are important to the proper modelling of this process.

The case study for the TPFF set was chosen to be the first case of the first fault which is a gradual closing of the air line valve while the air and water flow rates are varied. The monitoring statistics for the different algorithms can be found in Figures 3.22 - 3.25. The challenge with this case and TPFF as a whole is to model a process with multiple operating conditions.

SFA, SSFA, and SPCA were insensitive and could not detect the anomalous behaviour until it became very apparent. SSFA actually decreased its T^2 value when the fault appeared and did not increase again until more than halfway through the fault. SFA did have some early detections, but it would be hard to differentiate these from the similar spikes encountered before the fault was introduced. In encompassing the different operating conditions, these models became too generic and could not differentiate between a normal and abnormal change. MSSFA had the best performance by being able to detect

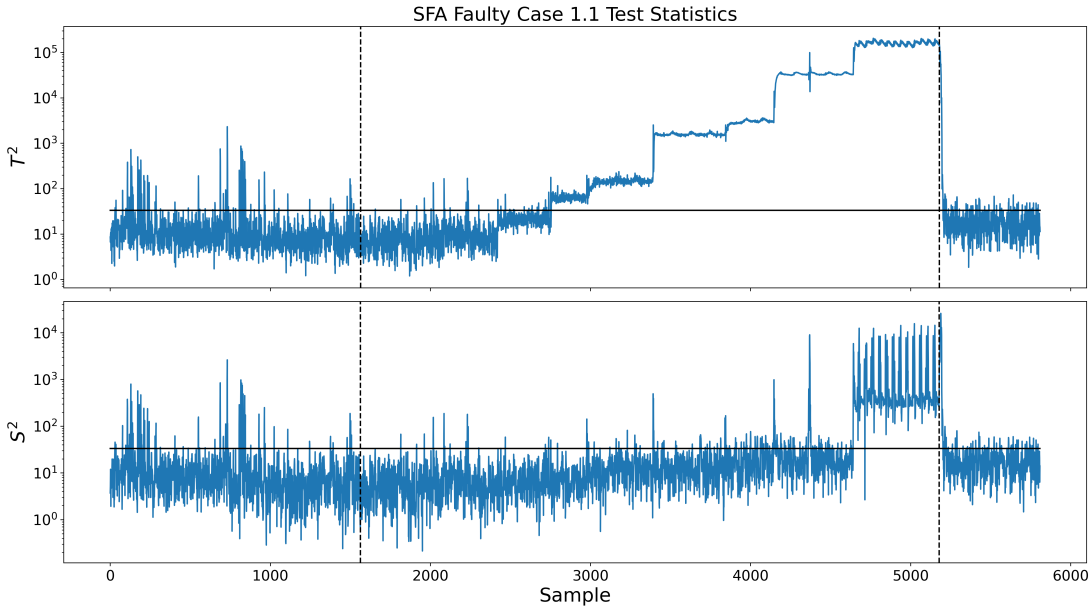

Figure 3.22: SFA monitoring statistics for case 1 of fault 1

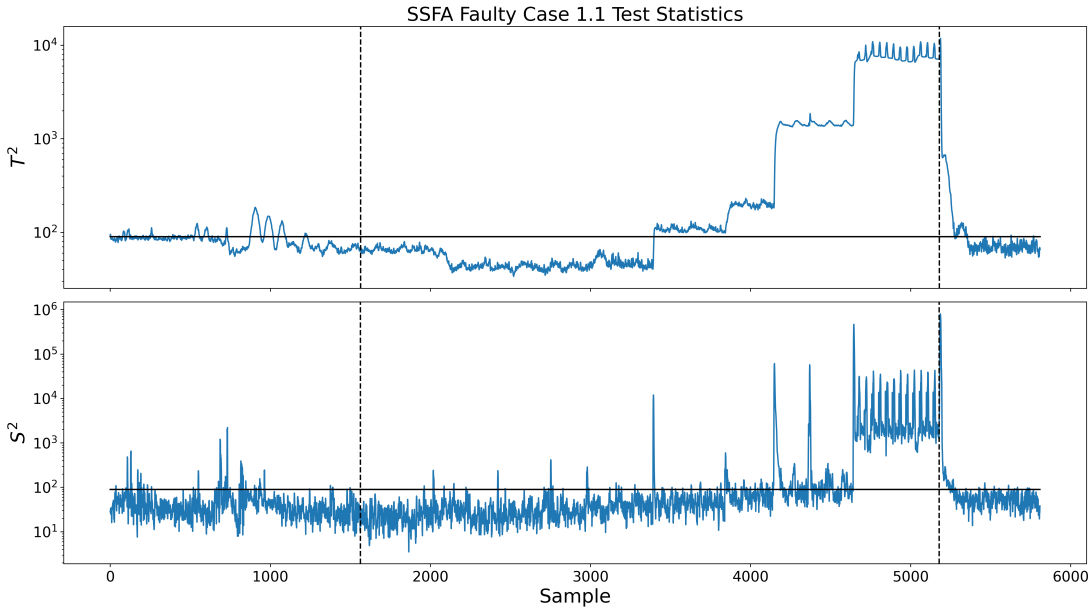

Figure 3.23: SSFA monitoring statistics for case 1 of fault 1

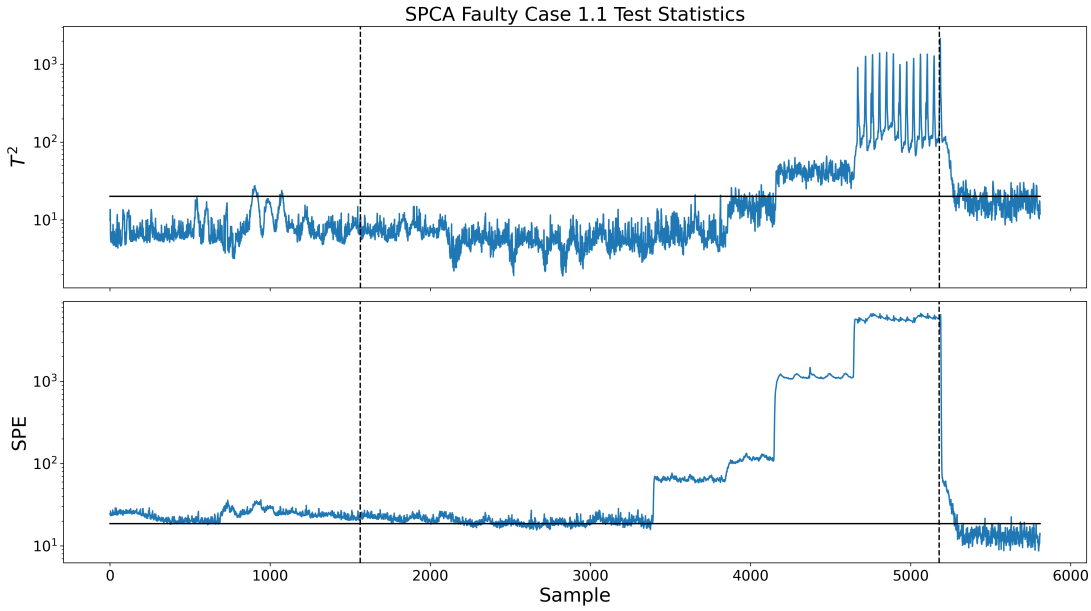

Figure 3.24: SPCA monitoring statistics for case 1 of fault 1

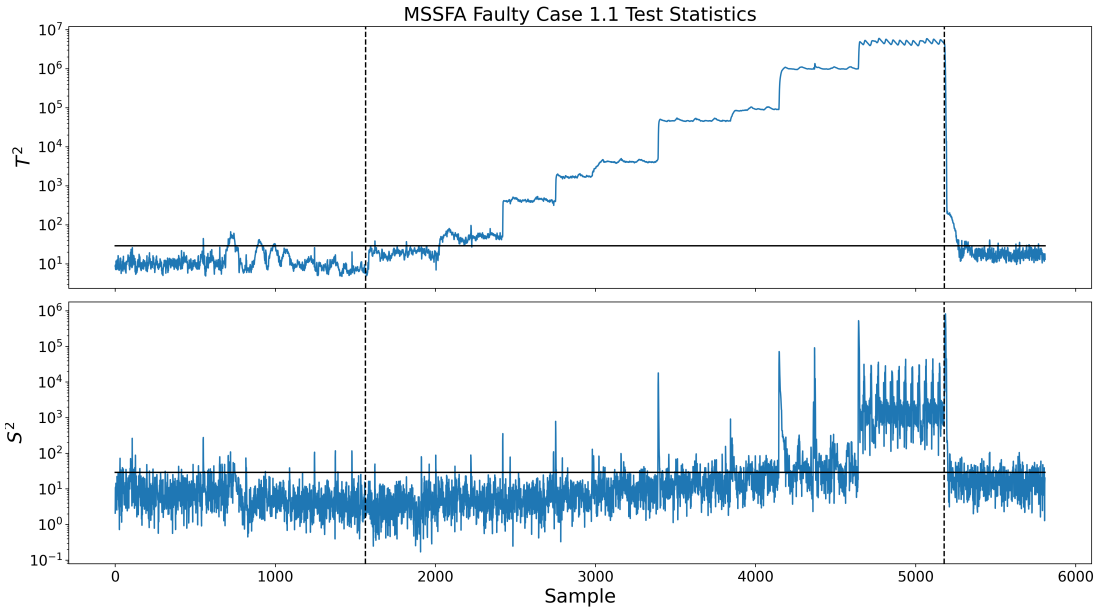

Figure 3.25: MSSFA monitoring statistics for case 1 of fault 1

Table 3.3: Comparison of FDRs and FARs for the batch algorithms using the TPFF dataset

	SFA		SSFA		SPCA		MSSFA	
Test Set	FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR
Fault 1								
Case 1	0.687	0.083	0.493	0.150	0.284	0.088	**0.876**	**0.069**
Case 2	0.700	**0.074**	0.421	0.200	0.308	0.234	**0.813**	0.203
Case 3	0.450	**0.046**	0.298	0.159	0.197	0.114	**0.704**	0.117
Fault 3								
Case 1	0.988	**0.057**	**0.997**	0.490	0.982	0.190	0.992	0.175
Case 2	0.565	**0.044**	0.989	0.652	0.906	0.727	**0.993**	0.682
Case 3	0.987	**0.073**	0.995	0.308	0.984	0.257	**0.996**	0.220
Fault 4								
Case 1	0.875	0.224	0.579	0.349	0.409	**0.048**	**0.919**	0.255
Case 2	0.727	**0.113**	0.201	0.137	0.175	0.217	**0.905**	0.284
Case 3	0.806	**0.171**	0.325	0.475	0.248	0.346	**0.898**	0.300
Fault 5								
Case 1	0.358	**0.664**	0.705	0.931	0.618	0.968	**1.000**	1.000
Case 2	0.418	**0.042**	**0.797**	0.454	0.516	0.167	0.764	0.124
Average	0.687	**0.145**	0.618	0.391	0.512	0.305	**0.896**	0.312

the fault earlier than all other models. The combined inclusion of sparsity and dynamics seemed to be well suited to this case.

3.5.2 Fault Diagnosis

The fault diagnosis results of case 1 of fault 1 was investigated and the results were plotted in Figures 3.26 - 3.29. The range around the removal of the fault was chosen for two reasons. The algorithms all detected this fault at different points and at this sample range, the T^2 values peaked, and all models are in agreement that a fault was present. Since the fault was present for a long period, any secondary effects were well established at this point meaning that the models must be able to distinguish these from the primary effects.

SFA pointed to the air delivery pressure and pressure at the bottom of the riser as the culprits which was the correct diagnosis. It also showed S^2 contributions from unrelated variables such as a level indicator and valve on the 3 phase separator. SSFA and MSSFA correctly diagnosed the problem without including any unrelated contributions. MSSFA

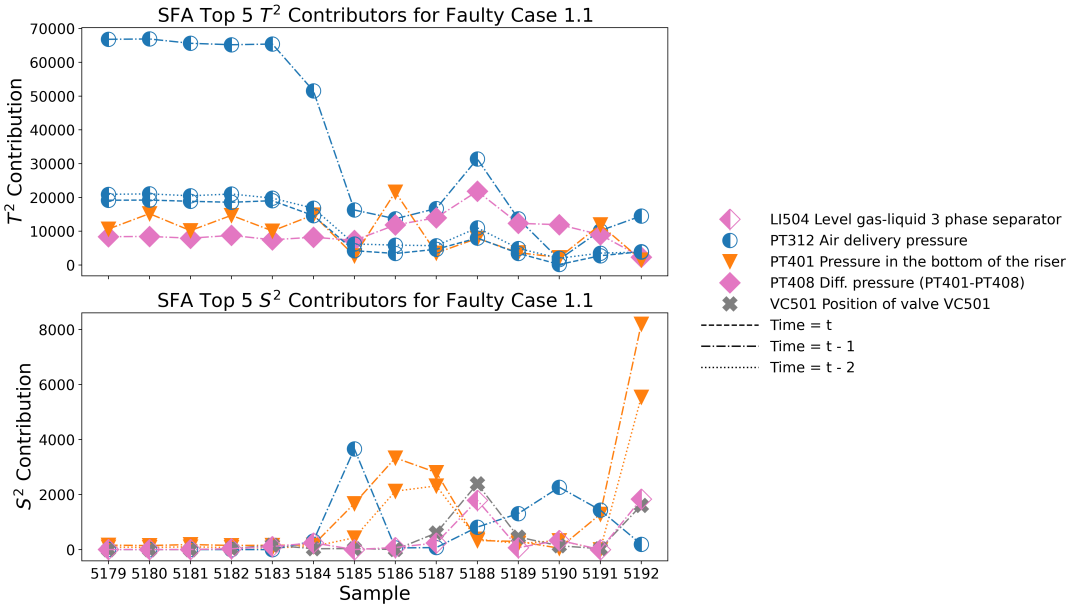

Figure 3.26: SFA fault diagnosis results for case 1 of fault 1

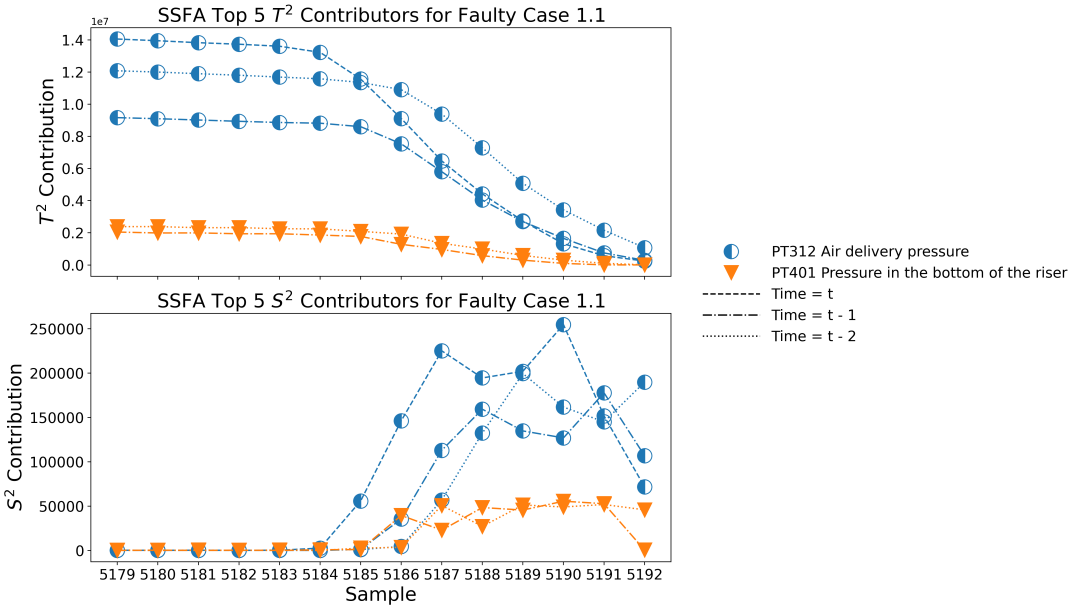

Figure 3.27: SSFA fault diagnosis results for case 1 of fault 1

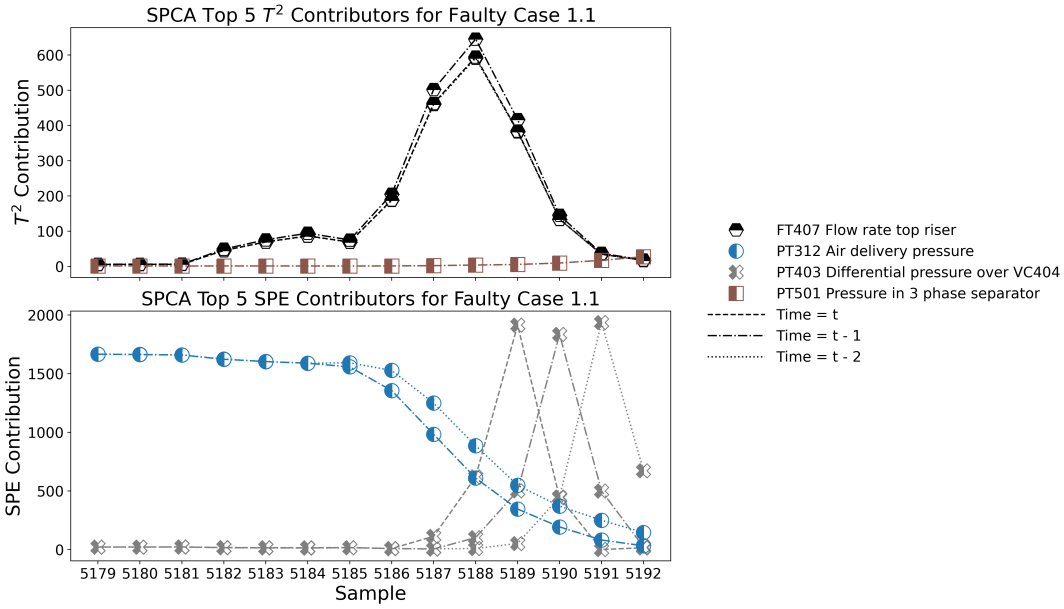

Figure 3.28: SPCA fault diagnosis results for case 1 of fault 1

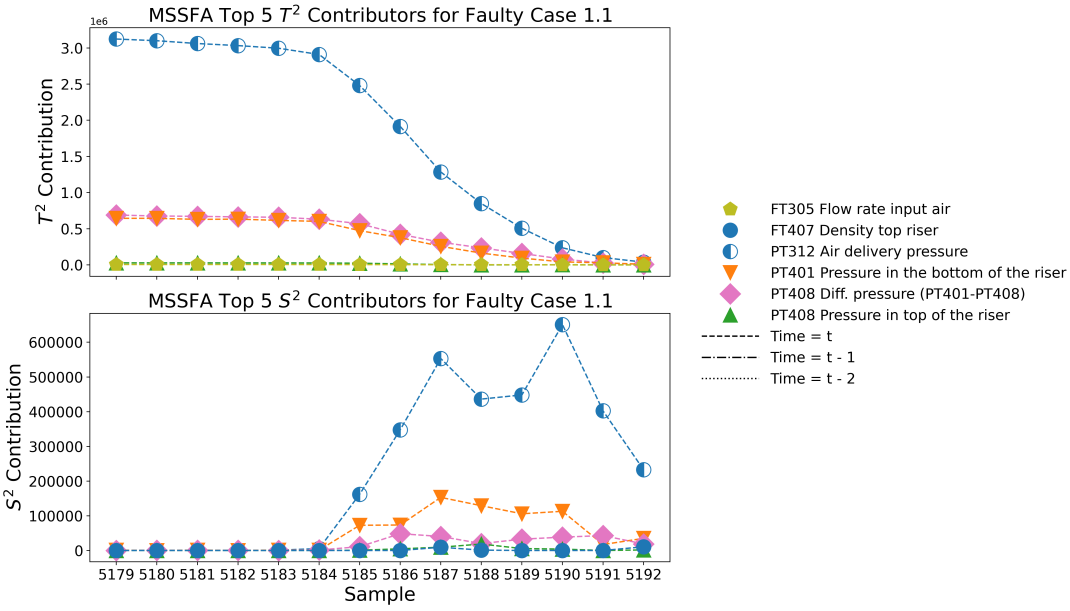

Figure 3.29: MSSFA fault diagnosis results for case 1 of fault 1

however managed to limit its diagnosis to only 3 variables, with the rest providing 0 contribution to the monitoring statistic. The SPCA SPE contributions did include the air delivery pressure but also the differential pressure on a valve far from the source of the fault at a similar magnitude. Its T^2 diagnosis was wrong as it pointed to the flow rate at the top of the riser as the culprit.

3.6 Conclusions

In this section, the effect of model sparsity was explored. A novel sparse SFA based algorithm called MSSFA was developed. This algorithm used proximal algorithms and manifold optimization to minimize the sparse SFA objective while adhering to its constraints. In addition to this algorithm, a fault detection and diagnosis framework was introduced. Using the TEP and TPFF data sets, MSSFA was compared with SFA, SSFA, and SPCA It was shown that MSSFA achieves improved interpretability and time complexity over the other algorithms, and was able to produce a more sparse model than SSFA. MSSFA also achieved superior monitoring and diagnosis performance as demonstrated by the TEP and TPFF case studies.

Chapter 4

Incremental Process Monitoring

4.1 IncSFA

IncSFA was developed for use on high dimensional visual inputs. Its linear complexity in terms on data dimensionality is attractive for process monitoring. This efficiency is useful for processes that have a lot of input variables or a high sample rate. It is achieved through the use of two covariance free algorithms: candid covariance-free incremental principal component analysis (CCIPCA) and covariance-free incremental minor component analysis (CIMCA).

4.1.1 Candid Covariance-Free Incremental PCA

The whitening matrix can be updated using CCIPCA [48] for each sample point. Conceptually, CCIPCA works by "pulling" the principal components into the direction of the sample. The magnitude of the projection of the sample onto the current principal component estimate determines the strength of this pull.

$$\mathbf{v}_i(t) = (1-\theta)\mathbf{v}_i(t-1) + \theta\left[\frac{\mathbf{u}_i(t)\cdot\mathbf{v}_i(t-1)}{||\mathbf{v}_i(t-1)||}\mathbf{u}_i(t)\right] \tag{4.1}$$

After this, the projection of the sample onto this principal component is subtracted from the sample.

$$\mathbf{u}_{i+1}(t) = \mathbf{u}_i(t) - \left[\frac{\mathbf{u}_i(t)\cdot\mathbf{v}_i(t)}{||\mathbf{v}_i(t)||}\right]\frac{\mathbf{v}_i(t)}{||\mathbf{v}_i(t)||} \tag{4.2}$$

This new sample is then used to extract the next principal component. Once all principal components are extracted, the whitening matrix can be calculated [21].

$$\mathbf{U} = \left[\frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}, \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|} \ldots, \frac{\mathbf{v}_K}{\|\mathbf{v}_K\|}\right] \tag{4.3}$$

$$\mathbf{S}_{i,j} = \begin{cases} \|\mathbf{v}_i\| & i = j \\ 0 & i \neq j \end{cases} \tag{4.4}$$

$$\mathbf{Q} = \mathbf{U}\mathbf{S}^{-1/2} \tag{4.5}$$

4.1.2 Covariance-Free Incremental Minor Component Analysis

CCIPCA cannot be used in place of the second SVD step in SFA as the directions of lowest variance are desired. Instead, CIMCA [21] can be used with the first derivative of the whitened input signals $\dot{\mathbf{z}}$. The vector $\mathbf{c}_i$ is used to move the current observation into a space where the previous minor components become principal components so that the next minor component can be extracted.

$$\mathbf{c}_1(t) = \mathbf{0} \in \mathbb{R}^K \tag{4.6}$$

$$\mathbf{c}_i(t) = \gamma \sum_{j=1}^{i-1} (\mathbf{p}_j(t) \cdot \mathbf{p}_i(t_{-1}))\mathbf{p}_j(t) \tag{4.7}$$

In this equation, γ is the largest eigenvalue of the derivative signals and can be found using one iteration of the CCIPCA algorithm. The columns of the transformation matrix $\mathbf{P}$ are then updated using the learning rate θ and normalized as follows.

$$\mathbf{a}_i(t) = (\dot{\mathbf{z}}(t)^\top \mathbf{p}_i(t-1))\dot{\mathbf{z}}(t) + \mathbf{c}_i(t) \tag{4.8}$$

$$\mathbf{p}_i(t) = \frac{(1-\theta)\mathbf{p}_i(t-1) - \theta\mathbf{a}_i(t)}{\|(1-\theta)\mathbf{p}_i(t-1) - \theta\mathbf{a}_i(t)\|} \tag{4.9}$$

4.2 IncSFA Based Process Monitoring

A process monitoring framework was developed for IncSFA. The framework includes the calculation of the monitoring statistics, control limits, and fault diagnosis contributions.

The first issue that arises in an incremental monitoring scheme is the calculation of J. This is resolved by extracting m features and then dynamically removing the fastest

$m - J$ features once J is calculated. For high dimensional data, $m^\star < m$ features can be extracted if J does not exceed $m^\star$. In the case where it does exceed it, J will be set to $m^\star$ which may degrade performance. After J is set, model updates can be made using only J features.

The calculation of J requires calculating the variance of $\dot{\mathbf{x}}$ and $\dot{\mathbf{y}}$ with Equation (2.9). The learning rate used in this chapter is

$$\theta = \max\left\{\frac{1}{k+2}, c\right\} \tag{4.10}$$

for update k and some constant minimum rate c. Setting this value too high will result in instability, and setting it too low will require more training data for convergence. Cross validation methods may be used to tune this parameter. For the investigations presented here, c was set to 10^{-4}. Using the learned variances, J can be found with Equation (2.14).

Once J has been chosen, calculating T^2 is straightforward.

$$T^2 = \mathbf{y}(t)^\top \mathbf{y}(t) \tag{4.11}$$

Calculating S^2 however requires $\mathbf{\Omega}$.

$$S^2 = \dot{\mathbf{y}}(t)^\top \mathbf{\Omega}^{-1} \dot{\mathbf{y}}(t) \tag{4.12}$$

This can be estimated as a matrix with the variances of $\dot{\mathbf{y}}$ along its diagonal and zeros everywhere else.

$$\mathbf{\Omega}_{i,j} = \begin{cases} s^2_{\dot{y}_i} & i = j \\ 0 & i \neq j \end{cases} \tag{4.13}$$

To calculate the control limits, n is taken as the number of samples the model was fed before it converged. The confidence level α is taken as a constant, although it would be simple to change α if desired as that would only mean calculating the control limits again.

$$\tau_{T^2} = \frac{J(n-1)(n+1)}{n(n-J)} F_\alpha(J, n-J) \tag{4.14}$$

$$\delta_{S^2} = \frac{J(n-2)n}{(n-1)(n-J-1)} F_\alpha(J, n-J-1) \tag{4.15}$$

Since the operation of finding the matrix square root is computationally expensive, the $\mathbf{M}$ matrices for fault diagnosis are only calculated if the model has converged or if a

training sample triggers an alarm. The signal contributions can then be found with the following equations.

$$\text{CDC}_{T^2,i} = \mathbf{x}^\top(\mathbf{W}\mathbf{W}^\top)^{\frac{1}{2}}\boldsymbol{\xi}_i\boldsymbol{\xi}_i^\top(\mathbf{W}\mathbf{W}^\top)^{\frac{1}{2}}\mathbf{x} \tag{4.16}$$

$$\text{CDC}_{S^2,i} = \dot{\mathbf{x}}^\top(\mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^\top)^{\frac{1}{2}}\boldsymbol{\xi}_i\boldsymbol{\xi}_i^\top(\mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^\top)^{\frac{1}{2}}\dot{\mathbf{x}} \tag{4.17}$$

The framework can be split into 2 parts. The first part is training where new samples are used to update the model, n, $\boldsymbol{\Omega}$, and the control limits. The second part is testing where the model is used to transform new samples, calculate T^2 and S^2, and perform fault diagnosis if necessary. Once the model has converged, only the testing part is required.

If T^2 has exceeded the control limit but S^2 has not, a slow drift anomaly has occurred. Essentially, the process has gradually changed its operating condition. If S^2 has exceeded the control limit but T^2 has not, a dynamic anomaly has occurred. This means that the dynamics of the process have changed without any significant change to the operating condition. When both of these statistics exceed their limits, an abrupt anomaly has occurred. This means that there has been a sudden change that caused a change in the operating condition.

Once any alarm has triggered, intervention by an engineer is required. In the case of a true fault, the relevant standard operating procedures would be performed before monitoring is resumed. In the case that a new normal operating condition has been encountered, model training would be resumed. A flowchart of the process monitoring scheme can be found in Figure 4.1.

4.3 Incremental MSSFA

The benefits of incremental algorithms has been discussed previously in Section 2.3. In practice, the incremental algorithms discussed are insufficient. The cubic complexity of RSFA updates prohibits its use for high dimensional data sets. For IncSFA, the complexity is ostensibly linear in terms of the input signals and quadratic in terms of the extracted features. However, when more than a few extracted features are required (as is typical for process monitoring applications), the CCIPCA algorithm suffers from stability issues and alternative methods that are more computationally expensive are required for whitening the data. This issue can be seen in Figure 6 of [48] where the correctness of eigenvectors starts to drop off after the first 5 are extracted.

An incremental algorithm was developed that does not suffer from these stability issues and lowers the complexity of updates in terms of the number of input signals from cubic to

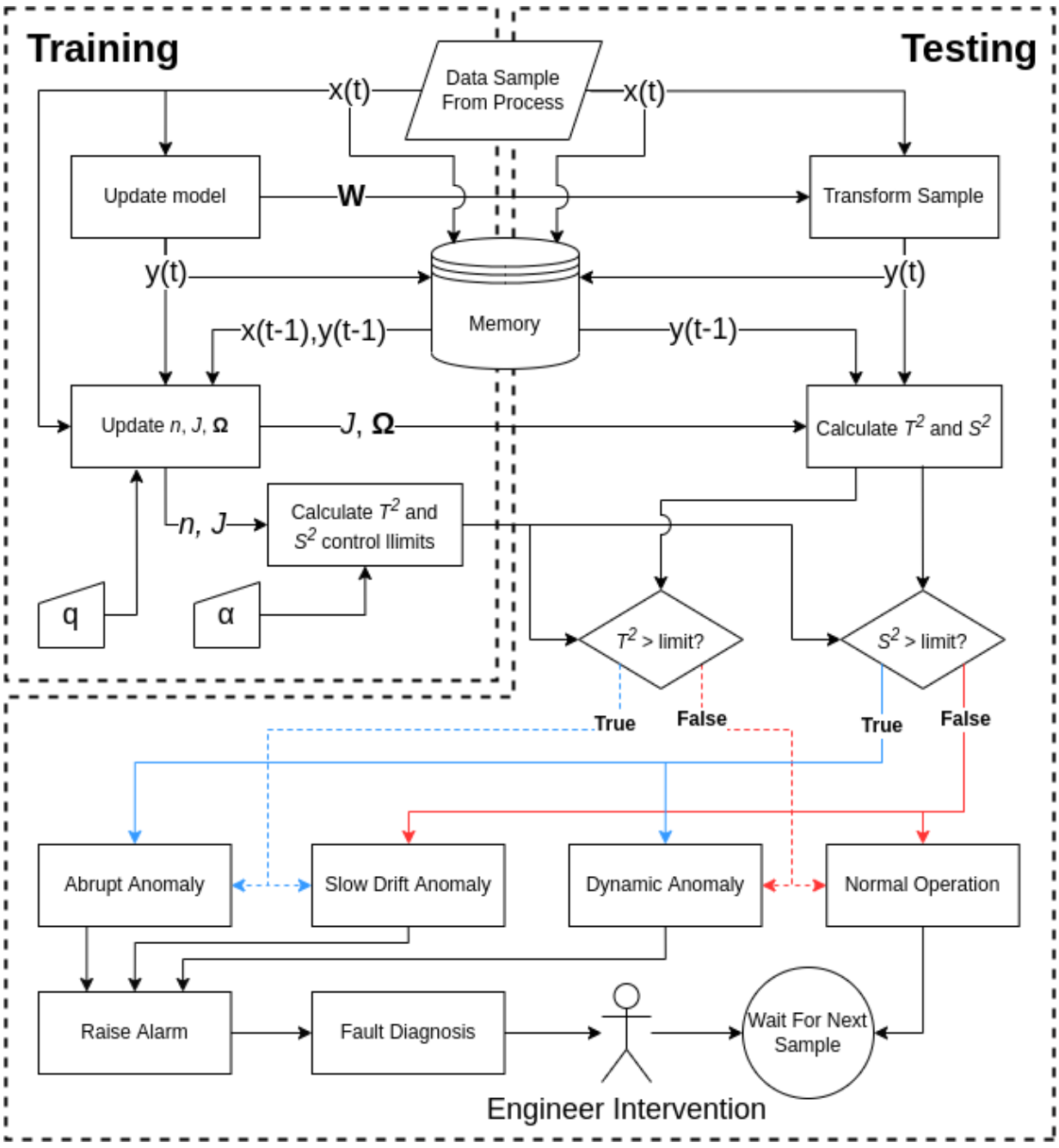

Figure 4.1: Incremental process monitoring framework

quadratic. In addition to this, this algorithm also provides a sparse model allowing for the simultaneous exploitation of sparsity and incremental updates. This algorithm is based on MSSFA and is called IncMSSFA.

IncMSSFA estimates the covariance matrices in the same manner as RSFA by using Equations (2.8) and (2.10). With each new sample, one iteration of MSSFA is performed. Each update has a time complexity of $\mathcal{O}(m^2 J)$ which is the cost of the retraction. This algorithm lacks the linear complexity of IncSFA, but the benefits of sparsity are more important for process data where the dimensionality is relatively low. The incremental monitoring scheme developed in Section 4.2 can be applied to this algorithm.

One issue that arises in the implementation of this algorithm is that the covariance matrix may not be positive definite especially for early samples which means that the retraction cannot be performed. This constraint arises from the Cholesky decomposition step. This is fixed by adding a small value ϵ to the diagonal of the matrix

$$\mathbf{A} \leftarrow \mathbf{A} + \epsilon \mathbf{I} \tag{4.18}$$

where $\mathbf{I}$ is the identity matrix. The value used for the investigations was 10^{-6}.

In addition to this change, the calculation of the step size α_k from MSSFA is replaced with $\alpha = 1$. Unlike in batch MSSFA, IncMSSFA is optimizing onto a moving target due to the covariance matrix updates. The removal of the diminishing step size allows the solution to move along with the target.

It may be beneficial to estimate the variance of the data as in Equation (2.9) to ensure the varying scales do not negatively impact the performance of the model. In the cases investigated, this did not have a significant effect on results, but it may be needed when the scales of the inputs vary more dramatically. A one-step update of IncMSSFA is described in Algorithm 7.

4.4 Tennessee Eastman Process Case Study

4.4.1 Sparsity & Interpretability

The sparsity of the incremental models was tested by extracting the first 55 features of the TEP data set. Additionally, the training data was normalized to prevent the variable scales from affecting the weighting of the variables.

Algorithm 7: IncMSSFA($\mathbf{W}, \mathbf{W}(t-1), \mathbf{x}^\star, \mathbf{x}(t-1), \bar{\mathbf{x}}, \mathbf{s}^2, \mathbf{A}, \mathbf{B}, \mathbf{y}(t-1), \mathbf{\Omega}, \theta, \epsilon$)

Data: $\mathbf{W} \in \mathbb{R}^{J\times m}, \mathbf{W}(t-1) \in \mathbb{R}^{J\times m}, \mathbf{x}^\star \in \mathbb{R}^m, \mathbf{x}(t-1) \in \mathbb{R}^m, \bar{\mathbf{x}} \in \mathbb{R}^m, \mathbf{s}^2 \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{m\times m}, \mathbf{B} \in \mathbb{R}^{m\times m}, \mathbf{y}(t-1) \in \mathbb{R}^J, \mathbf{\Omega} \in \mathbb{R}^{J\times J}, k \in \mathbb{N}, \theta \in \mathbb{R} > 0, \epsilon \in \mathbb{R} > 0$

Result: $\mathbf{W} \in \mathbb{R}^{m\times J}, \mathbf{x} \in \mathbb{R}^m, \bar{\mathbf{x}} \in \mathbb{R}^m, \mathbf{s}^2 \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{m\times m}, \mathbf{B} \in \mathbb{R}^{m\times m}, \mathbf{y} \in \mathbb{R}^J, \mathbf{\Omega} \in \mathbb{R}^{J\times J}$

$\bar{\mathbf{x}} \leftarrow (1-\theta)\bar{\mathbf{x}} + \theta\mathbf{x}^\star$;

$\mathbf{s}^2 \leftarrow (1-\theta)\mathbf{s}^2 + \theta(\mathbf{x}^\star - \bar{\mathbf{x}})^2$;

$\mathbf{x} \leftarrow \frac{\mathbf{x}^\star - \bar{\mathbf{x}}}{\mathbf{s}}$;

$\dot{\mathbf{x}} \leftarrow \mathbf{x} - \mathbf{x}(t-1)$;

$\mathbf{A} \leftarrow (1-\theta)\mathbf{A} + \theta\mathbf{x}\mathbf{x}^\top$;

$\mathbf{B} \leftarrow (1-\theta)\mathbf{B} + \theta\dot{\mathbf{x}}\dot{\mathbf{x}}^\top$;

$L \leftarrow 2||\mathbf{B}||_F$;

$\mathbf{V} \leftarrow \mathbf{W} + \frac{k}{k+3}(\mathbf{W} - \mathbf{W}(t-1))$;

$\boldsymbol{\eta} \leftarrow -\frac{2}{L}\mathbf{B}\mathbf{W}$;

$\mathbf{U} \leftarrow \text{CholeskyQRRetraction}(\mathbf{A} + \epsilon\mathbf{I}^{m\times m}, \mathbf{V}, \boldsymbol{\eta})$;

`/* sgn, max, and` $|\cdot|$ `are element-wise operations */`;

$\mathbf{W} \leftarrow \text{sgn}(\mathbf{U}) \max\{|\mathbf{U}| - \frac{1}{L}\mathbf{I}^{J\times m}, \mathbf{0}^{J\times m}\}$;

$\mathbf{y} \leftarrow \mathbf{W}^\top\mathbf{x}$;

$\dot{\mathbf{y}} \leftarrow \mathbf{y} - \mathbf{y}(t-1)$;

$\mathbf{\Omega} \leftarrow (1-\theta)\mathbf{\Omega} + \theta\text{diag}(\dot{\mathbf{y}}^2)$;

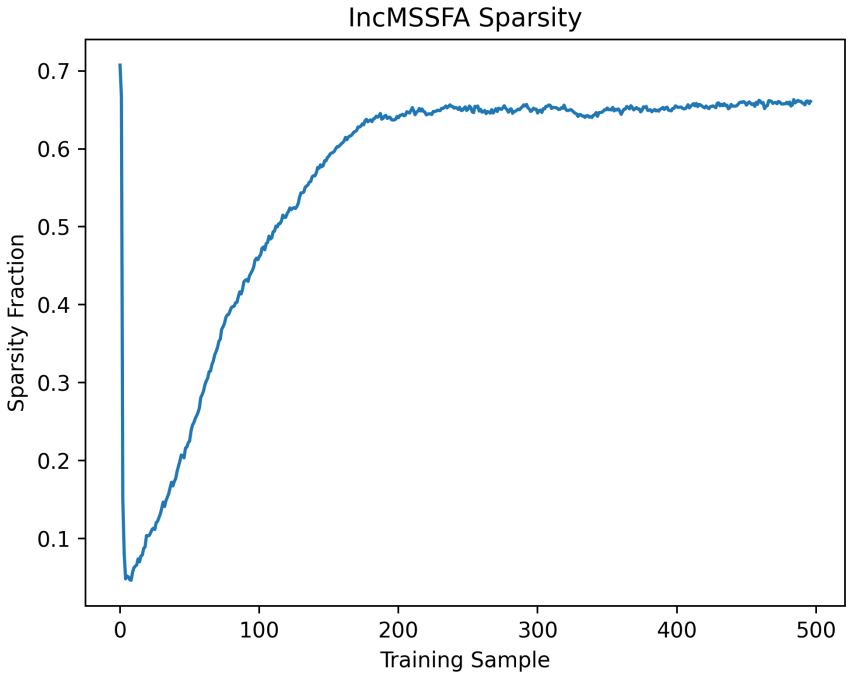

Figure 4.2: Sparsity fraction of IncMSSFA during training

The sparsity was calculated using Equation (3.24). As expected, both RSFA and IncSFA had a sparsity fraction of 0. IncMSSFA had a sparsity fraction of 0.661 and the sparsity fraction over the course of training is plotted in Figure 4.2. The sparsity drops off rapidly as the solution moves away from the initial guess of the identity matrix and then increases until it plateaus.

A visualization of the transformation matrices was plotted in Figure 4.3. The white spaces in this figure denote sparse values. RSFA and IncMSSFA values are well distributed across the matrix.

IncSFA had interesting results with most of the features being approximately equal to each other. This may arise from the use of CCIPCA as it tends to become increasingly unstable as the number of extracted components increase. This instability is due to a step in the algorithm where the projection of the sample onto the previous components is subtracted from the sample. The CCIPCA step can be replaced with an incremental estimation of the covariance matrix and a SVD step. This new algorithm will be referred to as IncSFA SVD. Using this algorithm solves the issues of stability but also removes

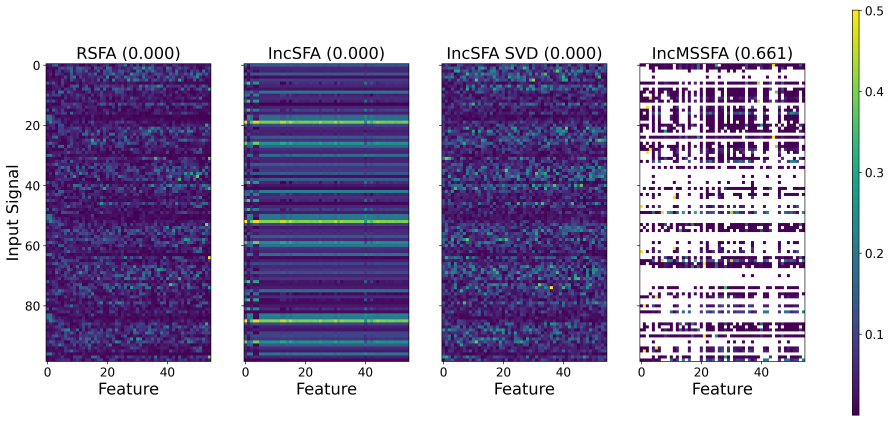

Figure 4.3: Transformation matrices of incremental algorithms

the linear complexity of the algorithm. The transformation matrix of IncSFA SVD is distributed more evenly and does not show any of the anomalous behaviour of IncSFA.

The top 10 contributors to the slowest feature were extracted to determine how interpretable the feature was. The slowest feature of RSFA may model the effect of manipulating the stripper steam valve. This can be seen as the 10 largest contributors which account for 34.76% of the feature are the stripper steam valve, steam flow, temperature, and pressure for the current and lagged samples. It is difficult to be sure about this interpretation because the reactor pressure is the 10th largest contributor and there is still 65.24% of contributions split across the rest of the variables.

IncSFA was similar in interpretability with the largest 10 contributors accounting for 36.32% of the feature. The feature seems to model the effect of the recycle stream on the separator pressure as the variables include the compressor work, recycle valve, and the separator pressure in the current and lagged samples. The interpretability of IncSFA SVD was actually worse for the slowest feature as the largest 10 contributors made up 27.31% of the total contributions. These variables were the flow rates of streams 2, 3, 4, and 10, the reactor temperature and cooling water flow, the compressor recycle valve, and the product separator underflow.

IncMSSFA is very easy to interpret as the stripper level and stripper liquid product flow for the first lagged sample contribute 49.18% and 49.17% respectively. The obvious interpretation here is that this feature models the effect that liquid flowing out of the

stripper has on the level of liquid inside the stripper.

4.4.2 Complexity & Runtime

This section details the computational complexity of the incremental algorithm updates. This complexity is given in terms of the number of input signals m and the number of desired output features J.

For RSFA, the algorithm has a complexity of $\mathcal{O}(m^3)$ which is due to the rank one modification step [22]. IncSFA complexity is $\mathcal{O}(m + J^2)$ with the quadratic complexity coming from the CIMCA algorithm [21]. When SVD is introduced, the complexity drastically increases to $\mathcal{O}(m^3 + J^2)$. The complexity of IncMSSFA is $\mathcal{O}(m^2 J)$ due to the manifold retraction.

The TEP training set was used to determine the runtimes of the incremental algorithms. In the first experiment, the number of lagged samples was varied so that the number of input signals m ranged from 33-330. This experiment extracts 10 features from the set. Another experiment was done with a fixed lag number of 2 resulting in 99 input signals. The number of extracted features J was varied from 9-99. Both experiments ran each algorithm 5 times and averaged the runtimes.

All algorithms were implemented in Python using the NumPy package [47] for matrix operations and the code was run on a 3.60GHz Intel i3-9100F CPU. The runtimes for the two experiments were plotted on logarithmic graphs in Figures 4.4 and 4.5.

From the first plot, it can be seen that RSFA and IncSFA SVD had the sharpest increases in runtime which was expected from the cubic complexity in terms of m. IncMSSFA also showed a less drastic nonlinear increase which corresponds to its quadratic complexity in terms of m. IncSFA runtime is fairly stable with increasing m values due to its linear complexity. It tends to be slower than IncMSSFA since the CIMCA and CCIPCA algorithms were implemented natively in Python while NumPy operations such as the Cholesky decomposition are vectorized and run using more efficient precompiled C code. Nevertheless, the trend is still apparent with IncSFA overtaking the other algorithms at higher m values. When the J values were varied, both IncSFA and IncSFA SVD showed the nonlinear increase expected from the quadratic complexity of the CIMCA algorithm. IncMSSFA had less dramatic increase owing to its linear complexity in terms of J. RSFA was fairly stable as its complexity does not depend on J.

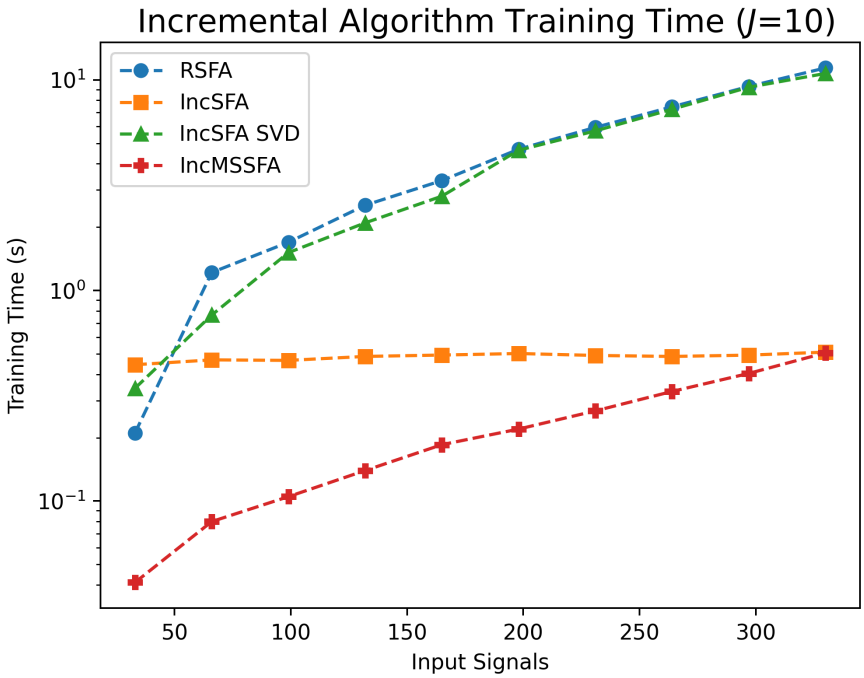

Figure 4.4: Training times of the incremental algorithms for a varying number of input signals and constant number of features extracted

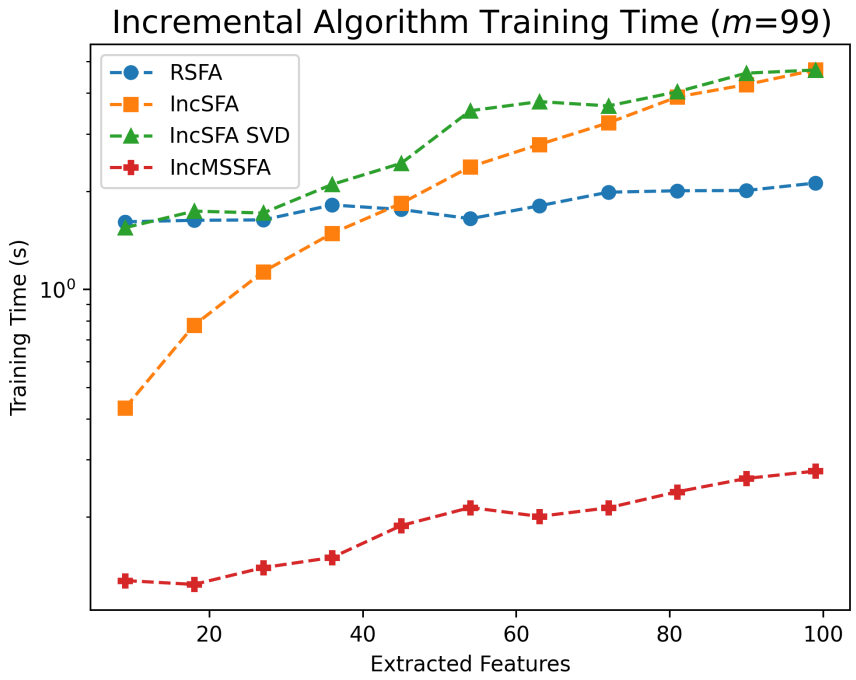

Figure 4.5: Training times of the incremental algorithms for a varying number of extracted features and constant number of input signals

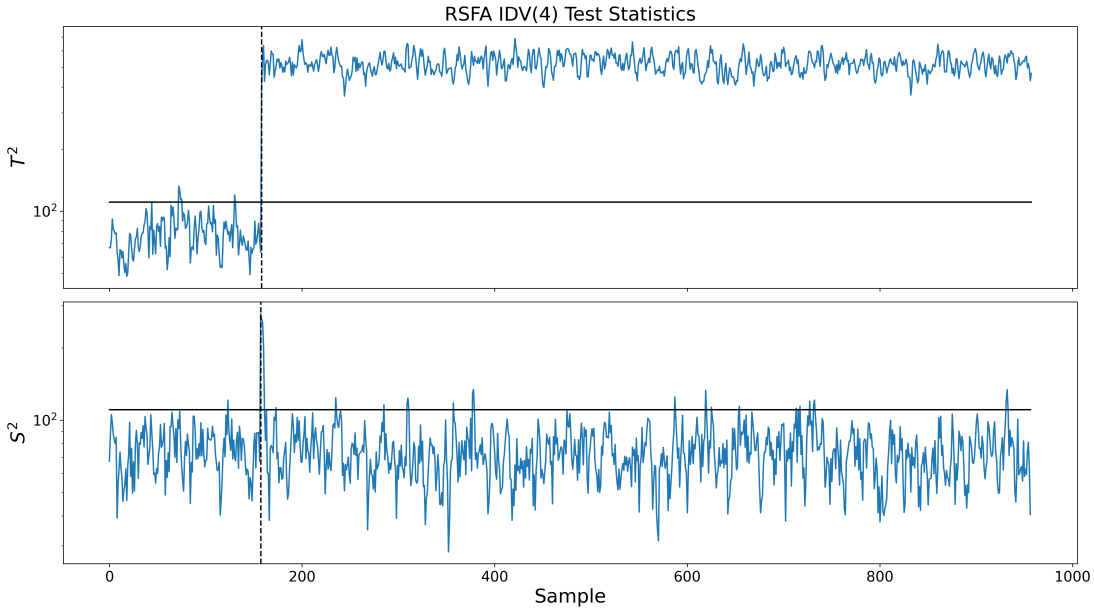

Figure 4.6: RSFA monitoring statistics for IDV(4)

4.4.3 Fault Detection

IDV(0) training data was used to train each algorithm and the FDR and FAR values were found using the T^2 monitoring statistic. Equation (2.14) was used to determine the J values of each algorithm. These values were found to be 64, 99, 89, and 86 for RSFA, IncSFA, IncSFA SVD, and IncMSSFA respectively. The monitoring results of these algorithms can be found in Table 4.1.

The results of IncSFA were insensitive to faults leading to poor FDR performance and FAR values of 0 for every test set. The addition of sparsity had a positive effect on monitoring performance with IncMSSFA having all FAR values except for IDV(0) and IDV(9) equal to 0 along with the best average FDR performance. IncSFA SVD outperformed RSFA in both FDR and FAR.

The monitoring statistics of IDV(4) and IDV(11) were plotted as a case study. The faults are a step change and random variations in the reactor cooling water inlet temperature. The monitoring statistic are plotted in Figures 4.6 - 4.9 for IDV(4) and Figures 4.10 - 4.13 for IDV(11). The solid horizontal black lines in these figures are the control limits and the vertical dashed lines indicate when the faults are introduced.

Except for IncSFA, the algorithms were able to detect the start of the fault for IDV(4).

Table 4.1: Comparison of FDRs and FARs for the incremental algorithms using the TEP dataset

	RSFA		IncSFA		IncSFA SVD		IncMSSFA	
Test Set	FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR
IDV(0)	0.000	0.041	0.000	**0.000**	0.000	0.045	0.000	0.001
IDV(1)	0.998	0.013	0.496	0.000	0.999	0.013	0.999	0.000
IDV(2)	0.985	0.013	0.088	0.000	0.985	0.013	0.985	0.000
IDV(3)	**0.064**	0.006	0.000	0.000	0.058	0.019	0.001	0.000
IDV(4)	1.000	0.038	0.000	0.000	1.000	0.019	1.000	0.000
IDV(5)	0.294	0.038	0.165	0.000	0.302	0.019	**1.000**	0.000
IDV(6)	0.994	0.006	0.991	0.000	1.000	0.006	1.000	0.000
IDV(7)	1.000	0.019	0.279	0.000	1.000	0.006	1.000	0.000
IDV(8)	0.980	0.019	0.639	0.000	0.980	0.013	0.979	0.000
IDV(9)	**0.071**	0.120	0.000	**0.000**	0.070	0.076	0.001	0.006
IDV(10)	0.759	0.025	0.116	0.000	0.810	0.038	**0.876**	0.000
IDV(11)	**0.940**	0.032	0.000	0.000	0.915	0.032	0.848	0.000
IDV(12)	0.992	0.006	0.720	0.000	0.992	0.032	**0.999**	0.000
IDV(13)	0.952	0.006	0.776	0.000	**0.958**	0.000	0.955	0.000
IDV(14)	1.000	0.019	0.000	0.000	1.000	0.019	1.000	0.000
IDV(15)	**0.091**	0.057	0.000	0.000	0.082	0.038	0.040	0.000
IDV(16)	0.552	0.139	0.049	0.000	0.868	0.108	**0.898**	0.000
IDV(17)	0.969	0.044	0.265	0.000	0.976	0.032	**0.978**	0.000
IDV(18)	**0.915**	0.038	0.870	0.000	0.914	0.044	0.905	0.000
IDV(19)	0.708	0.013	0.000	0.000	0.974	0.000	**0.980**	0.000
IDV(20)	0.718	0.013	0.202	0.000	0.814	0.013	**0.905**	0.000
IDV(21)	0.485	0.038	0.071	0.000	**0.486**	0.051	0.469	0.000
Average	0.737	0.034	0.273	0.000	0.771	0.029	**0.801**	0.000

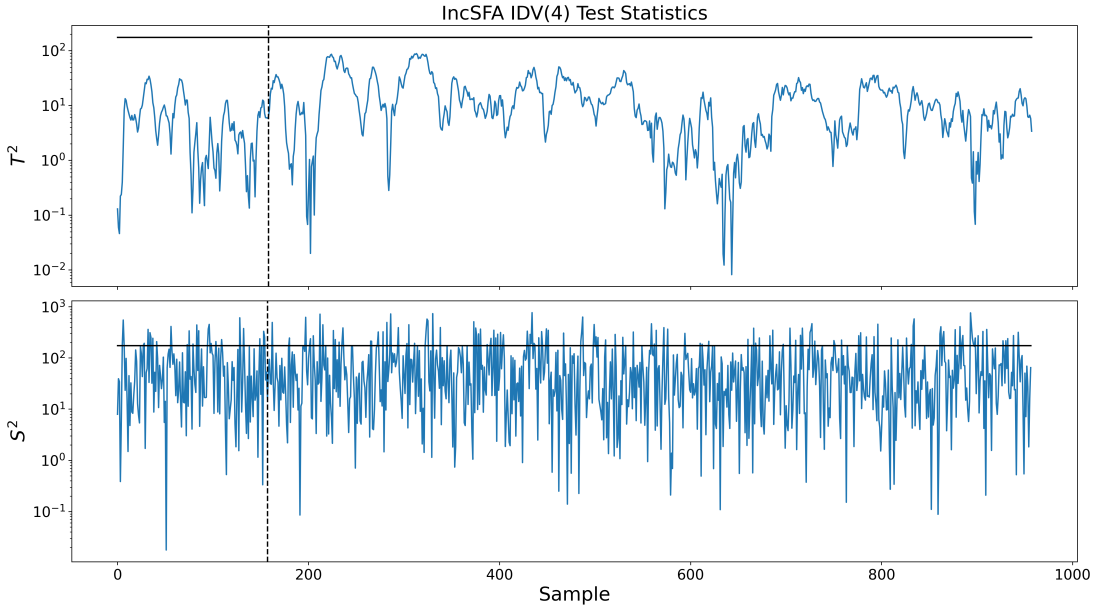

Figure 4.7: IncSFA monitoring statistics for IDV(4)

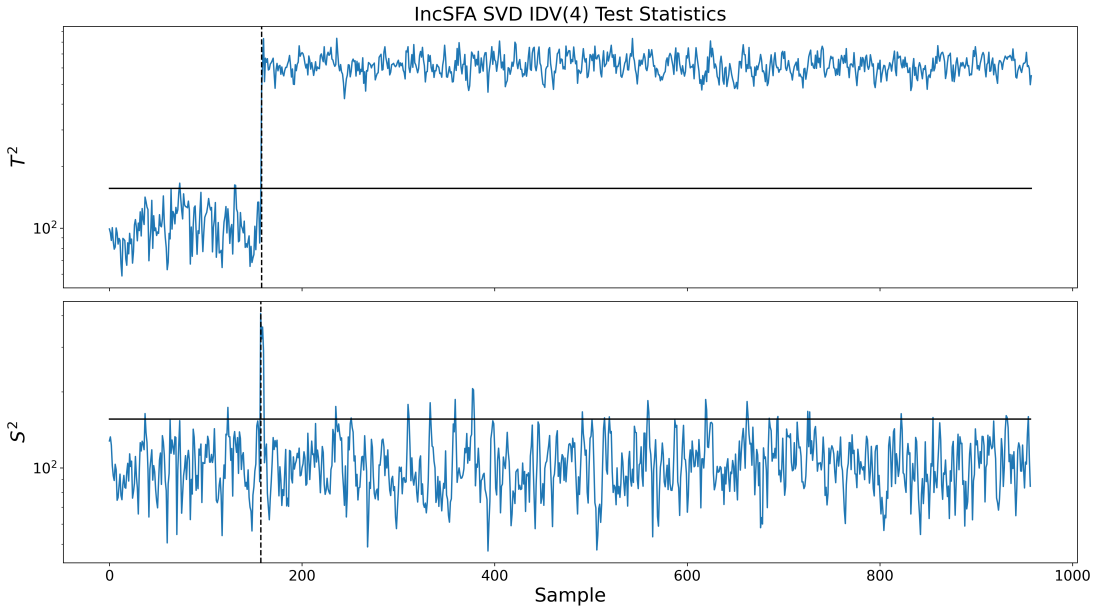

Figure 4.8: IncSFA SVD monitoring statistics for IDV(4)

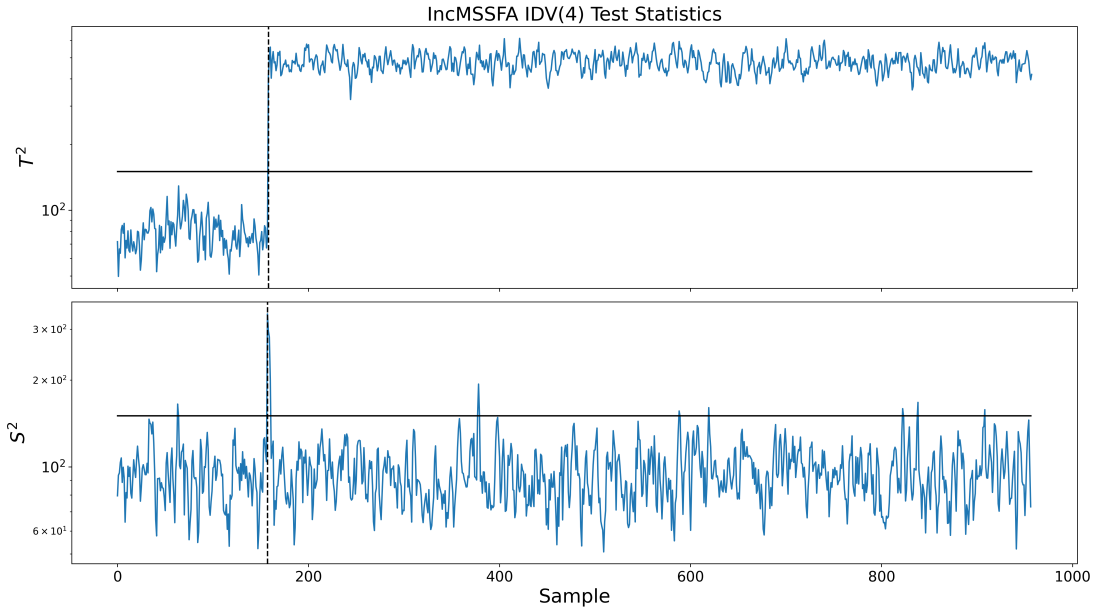

Figure 4.9: IncMSSFA monitoring statistics for IDV(4)

False alarms can be seen in RSFA and IncSFA SVD as T^2 values exceeded the control limit before the introduction of the fault. IncMSSFA did not have these false alarms and kept its T^2 value far below the limit before the introduction of the fault.

Similar performance was observed with IDV(11). IncSFA was again unable to detect the fault and IncMSSFA lacked the false alarms present in both RSFA and IncSFA SVD.

4.4.4 Fault Diagnosis

The case studies above are continued here to assess the fault diagnosis performance of the incremental algorithms. The CDC contributions were calculated, and the top 5 contributors over a given range were plotted. The range for IDV(4) coincides with the introduction of the fault.

The fault diagnosis plots are located in Figures 4.14 - 4.17 for IDV(4). Except for IncSFA, the algorithms were able to correctly diagnose the result by pointing to the reactor temperature and reactor cooling water flow as the main culprits for the fault. IncSFA SVD and IncMSSFA showed contributions from the reactor cooling water outlet temperature, which is related to fault, to their S^2 values.

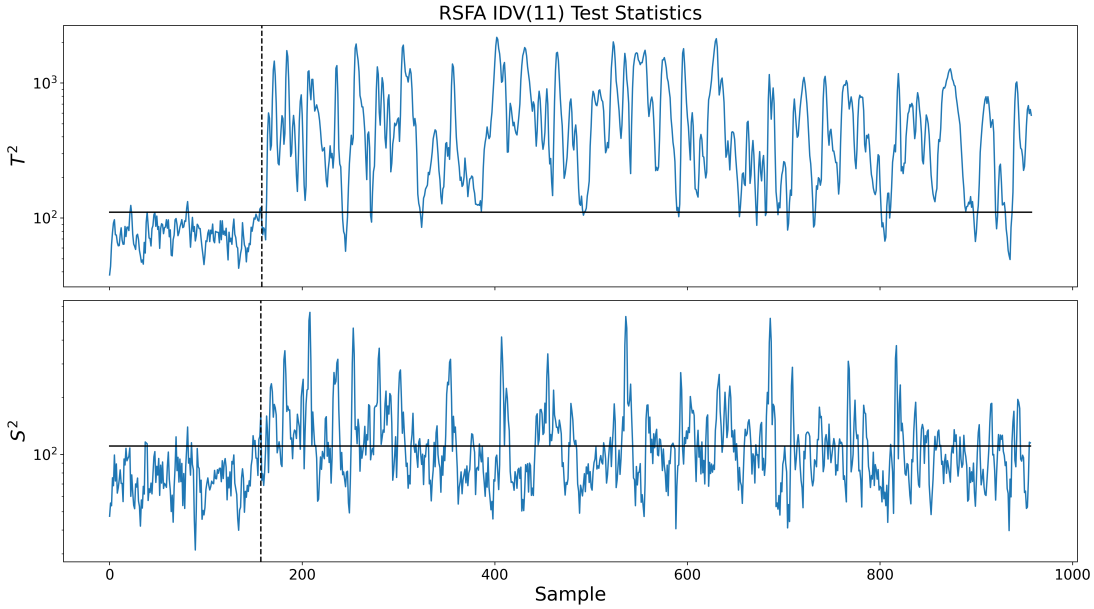

Figure 4.10: RSFA monitoring statistics for IDV(11)

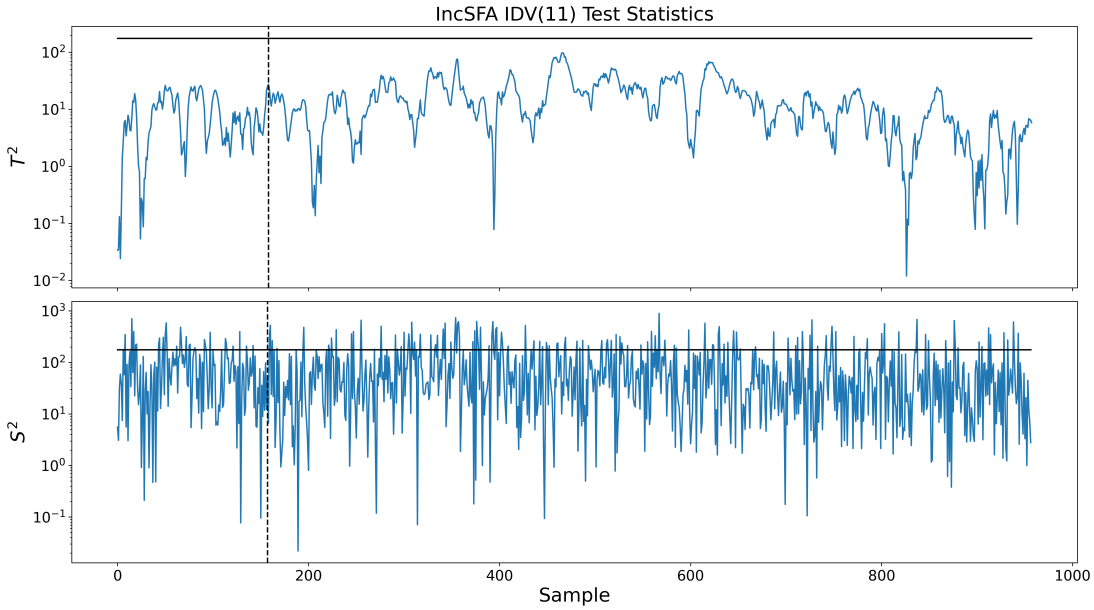

Figure 4.11: IncSFA monitoring statistics for IDV(11)

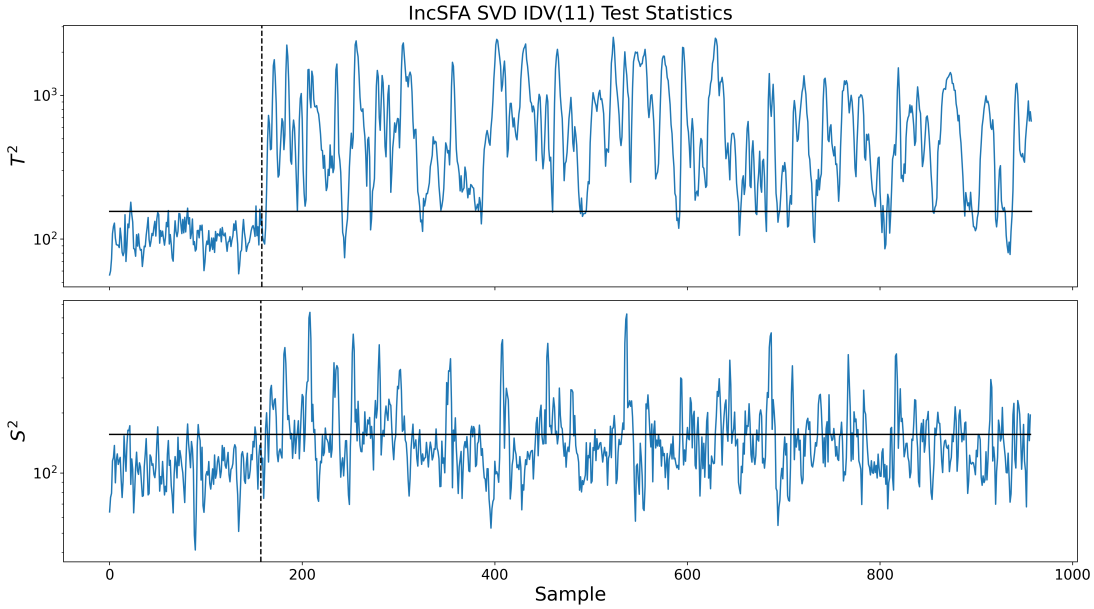

Figure 4.12: IncSFA SVD monitoring statistics for IDV(11)

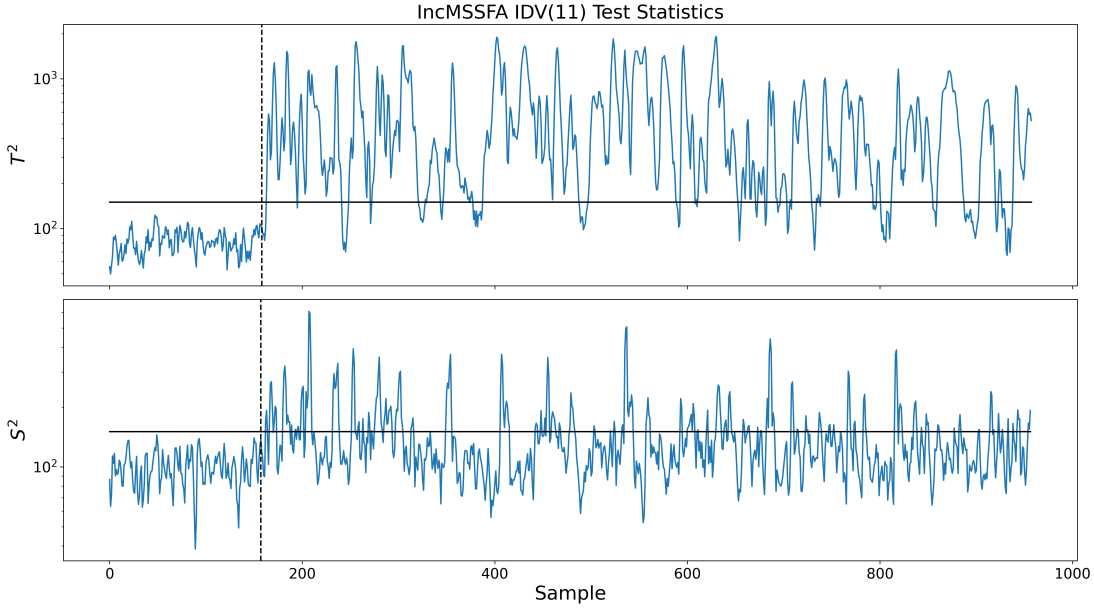

Figure 4.13: IncMSSFA monitoring statistics for IDV(11)

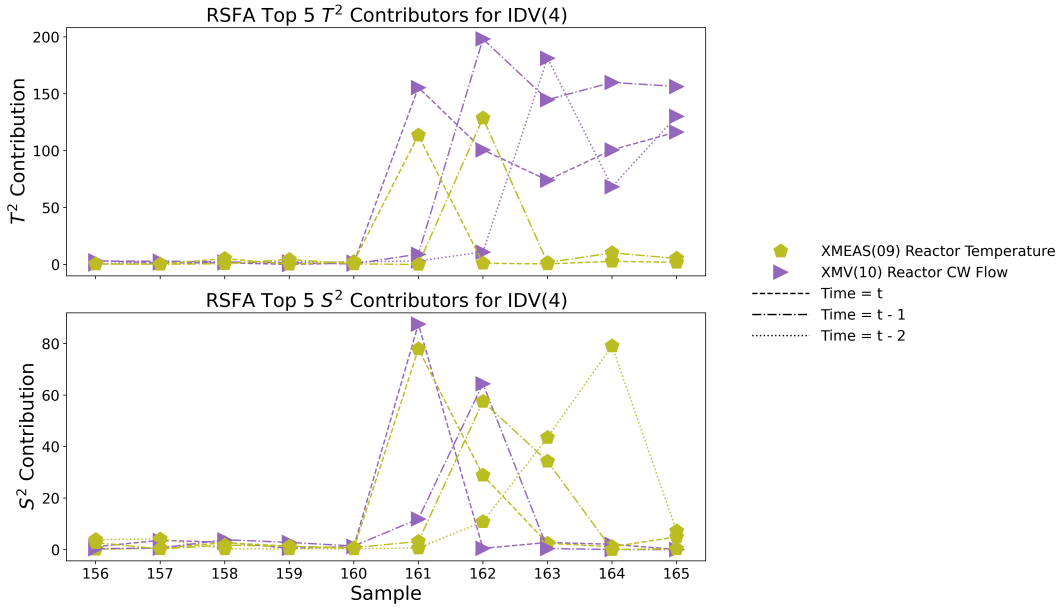

Figure 4.14: RSFA fault diagnosis results for IDV(4)

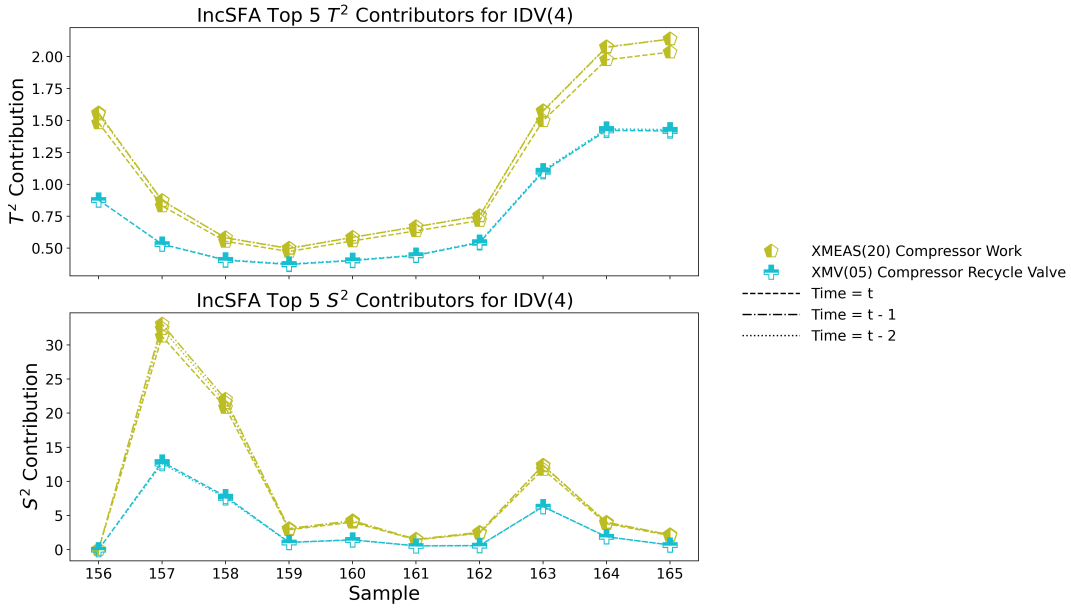

Figure 4.15: IncSFA fault diagnosis results for IDV(4)

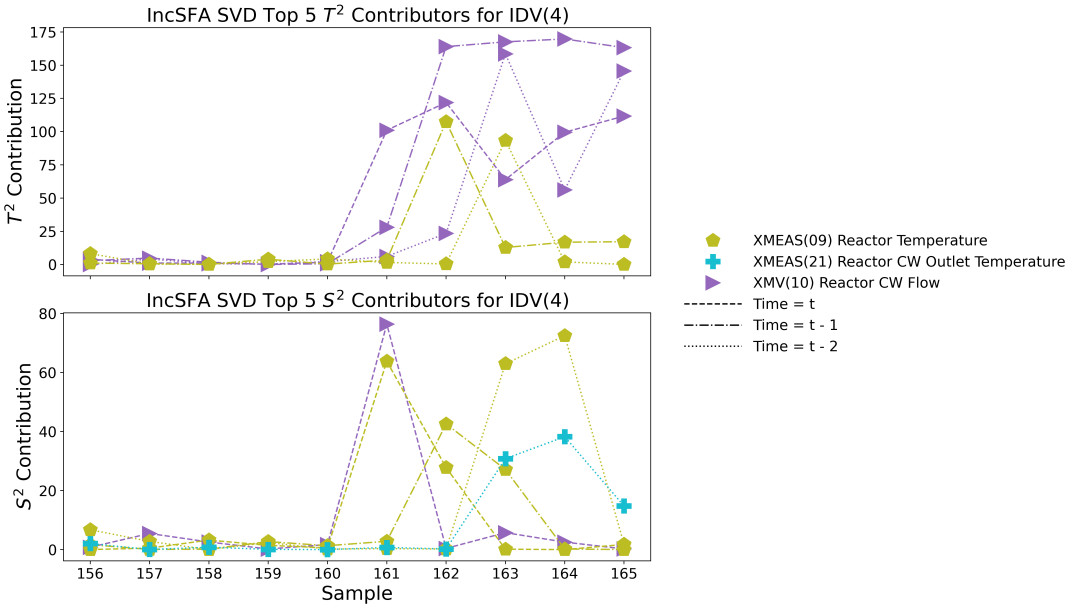

Figure 4.16: IncSFA SVD fault diagnosis results for IDV(4)

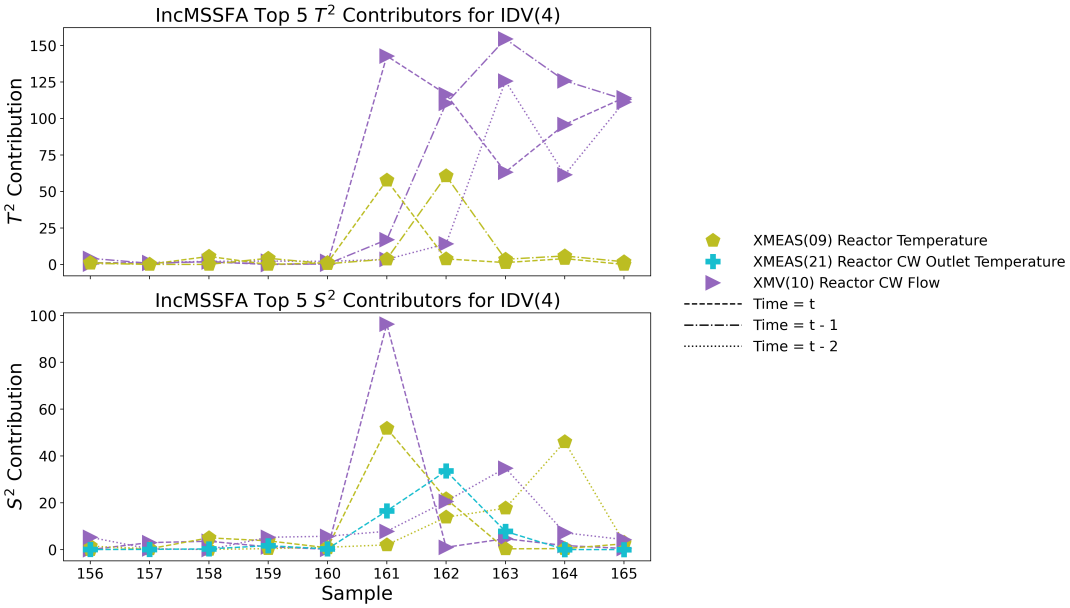

Figure 4.17: IncMSSFA fault diagnosis results for IDV(4)

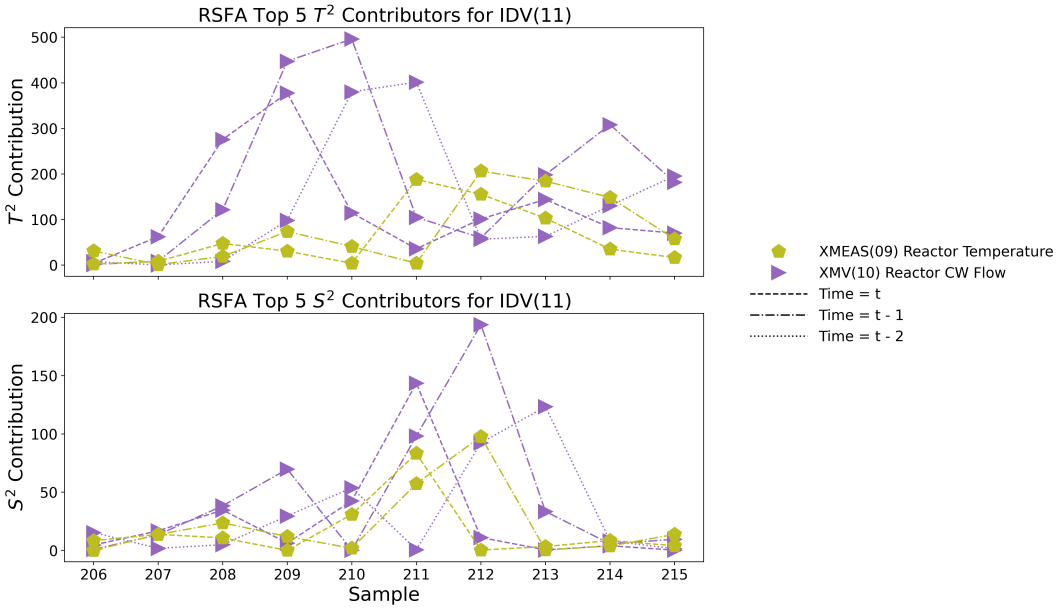

Figure 4.18: RSFA fault diagnosis results for IDV(11)

The IDV(11) diagnosis performance was the same as the IDV(4) performance. The contributions can be found in Figures 4.18 - 4.21. The range chosen here was around sample 210 to allow the random variations to affect the whole system. Any secondary effects present after 50 samples did not affect the performance of these algorithms.

4.5 Three Phase Flow Facility Case Study

4.5.1 Fault Detection

Training sets 2 and 3 of the TPFF data were used to train the incremental algorithms. This was also done in the original paper by [41]. Table 4.2 shows the monitoring results of the incremental algorithms.

For these algorithms, IncSFA showed the worst performance as it labelled almost every sample as faulty. When this is excluded, IncMSSFA had the best average FDR along with the best FDR in almost every set. IncSFA SVD had the lowest FAR values for almost every set but its average FDR was the worst out of the 4 incremental algorithms. RSFA

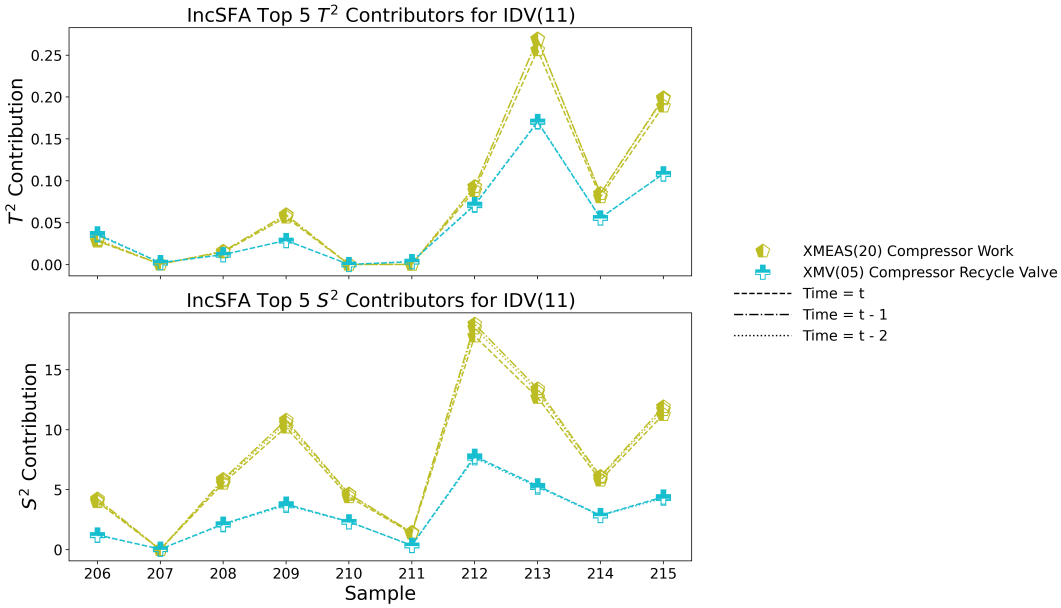

Figure 4.19: IncSFA fault diagnosis results for IDV(11)

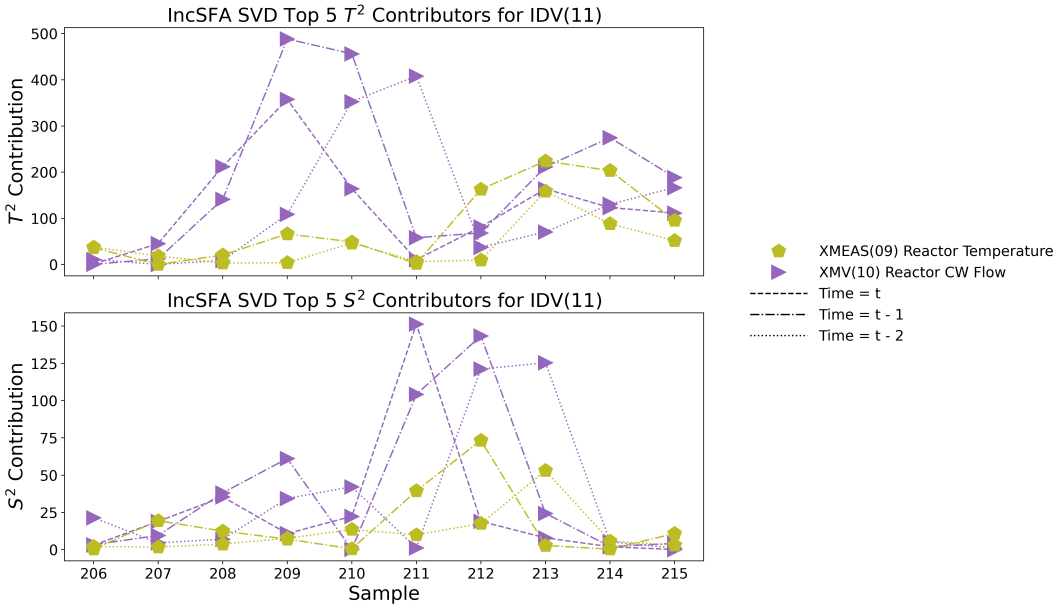

Figure 4.20: IncSFA SVD fault diagnosis results for IDV(11)

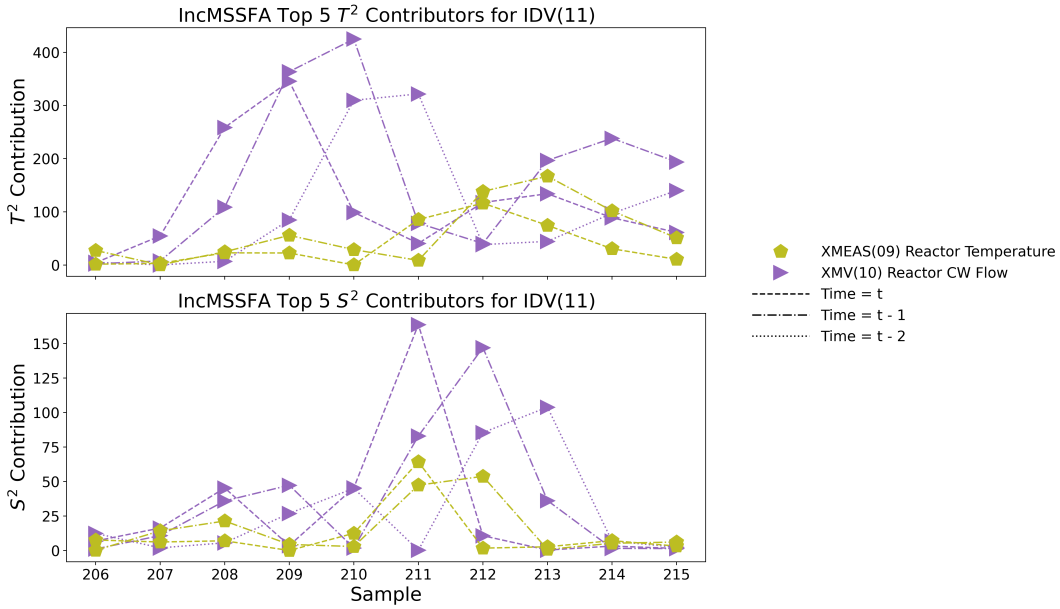

Figure 4.21: IncMSSFA fault diagnosis results for IDV(11)

had a larger average FDR and FAR than IncSFA SVD but did not come close to the FDR of IncMSSFA.

Case 1 of fault 1 was used as a case study. This fault case is a gradual closing of the air line valve with varying air and water flow rates. The monitoring statistic for the different algorithms can be found in Figures 4.22 - 4.25. Since both the training and test set each contain multiple operating conditions, it is important for the algorithm to accurately model the process instead of providing the model that best fits the training data.

All the models were able to detect the fault eventually as it grew large enough. RSFA and IncSFA SVD detected it very late. IncSFA was too sensitive and labelled all the data as faulty. In trying to fit itself to the changing operating conditions as it encountered them during training, this model was unable to establish what constituted a normal operating condition. IncMSSFA had a few false alarms before the fault was introduced, but it was able to detect the fault almost immediately after its introduction.

Figure 4.22: RSFA monitoring statistics for case 1 of fault 1

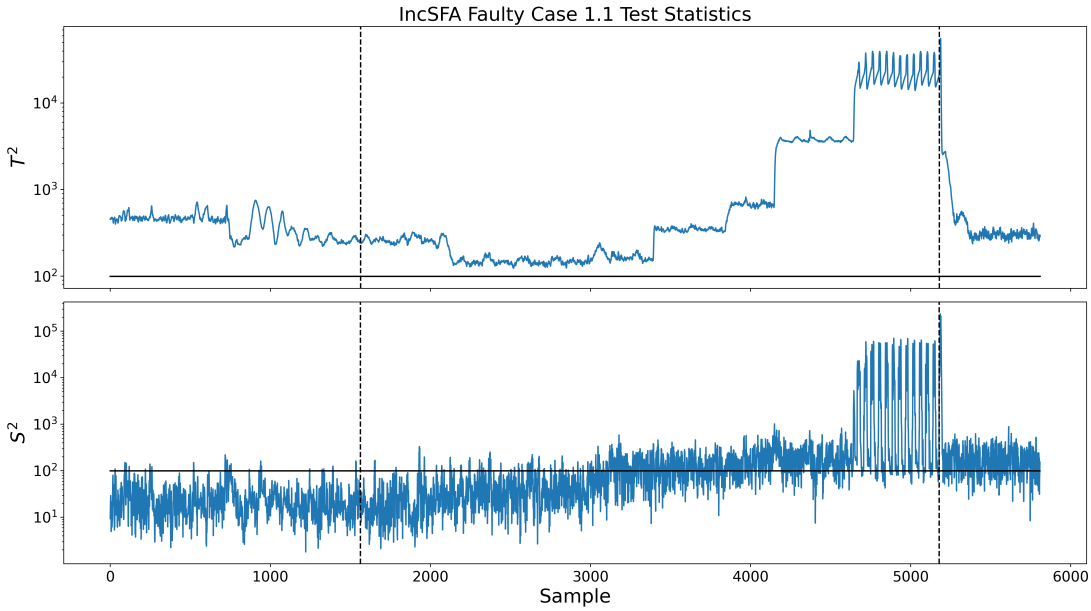

Figure 4.23: IncSFA monitoring statistics for case 1 of fault 1

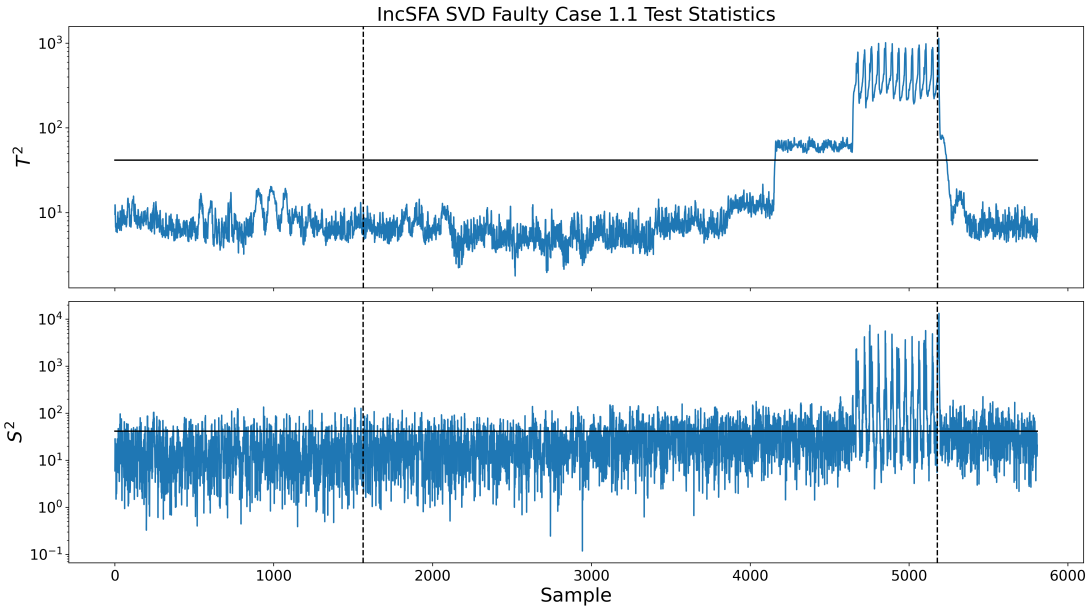

Figure 4.24: IncSFA SVD monitoring statistics for case 1 of fault 1

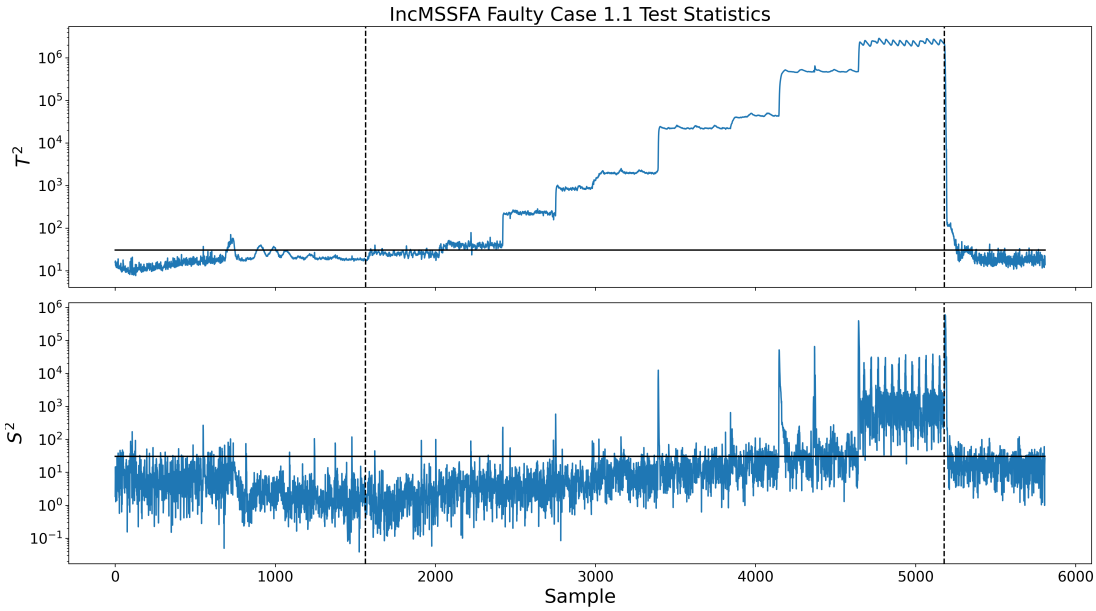

Figure 4.25: IncMSSFA monitoring statistics for case 1 of fault 1

Table 4.2: Comparison of FDRs and FARs for the incremental algorithms using the TPFF dataset

	RSFA		IncSFA		IncSFA SVD		IncMSSFA	
Test Set	FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR
Fault 1								
Case 1	0.303	**0.023**	**1.000**	1.000	0.266	0.026	0.780	0.058
Case 2	0.325	0.088	**1.000**	1.000	0.303	**0.080**	0.806	0.111
Case 3	0.203	0.067	**1.000**	1.000	0.197	**0.063**	0.594	0.069
Fault 3								
Case 1	0.987	0.077	**1.000**	0.998	0.987	**0.059**	0.991	0.112
Case 2	0.677	0.239	**1.000**	1.000	0.699	**0.160**	0.693	0.388
Case 3	0.993	0.092	**1.000**	1.000	0.993	**0.073**	0.993	0.148
Fault 4								
Case 1	0.529	0.022	**1.000**	1.000	0.415	**0.018**	0.918	0.237
Case 2	0.218	0.070	**1.000**	1.000	0.146	**0.069**	0.887	0.080
Case 3	0.290	0.167	**1.000**	1.000	0.223	**0.155**	0.893	0.190
Fault 5								
Case 1	0.543	0.764	**1.000**	1.000	0.407	**0.733**	0.613	0.906
Case 2	0.457	0.007	**1.000**	1.000	0.339	**0.000**	0.623	0.025
Average	0.502	0.147	**1.000**	1.000	0.452	**0.131**	0.799	0.211

4.5.2 Fault Diagnosis

The case study above is continued here by assessing the fault diagnosis capability of the incremental algorithms. The diagnosis range was chosen to be at the removal of the fault since this allowed the fault to establish secondary effects across the system forcing the models to distinguish these from the primary effects. The diagnosis plots are found in Figures 4.26 - 4.29.

IncMSSFA was the only incremental algorithm that correctly diagnosed this fault showcasing the benefits of sparsity. IncMSSFA even limited its diagnosis to 2 variables for T^2. For S^2, 2 other variables that were related to the fault but further from the source provided negligible contributions. The other models provided incorrect diagnosis results pointing to the differential pressure on a valve connected to the 2 phase separator along with the density, pressure, and flow rate at the top of the riser.

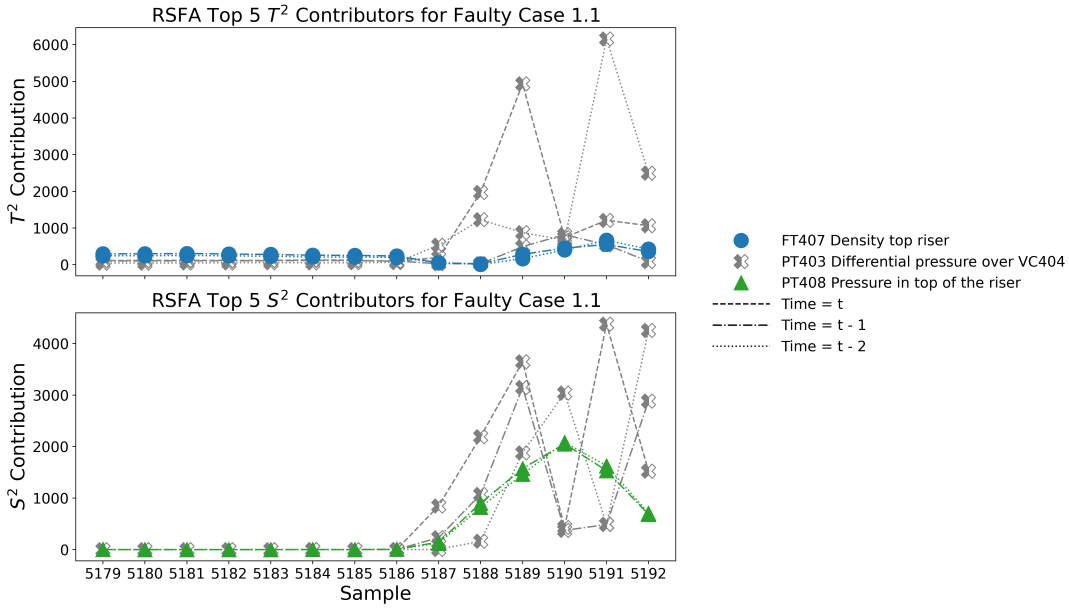

Figure 4.26: RSFA fault diagnosis results for case 1 of fault 1

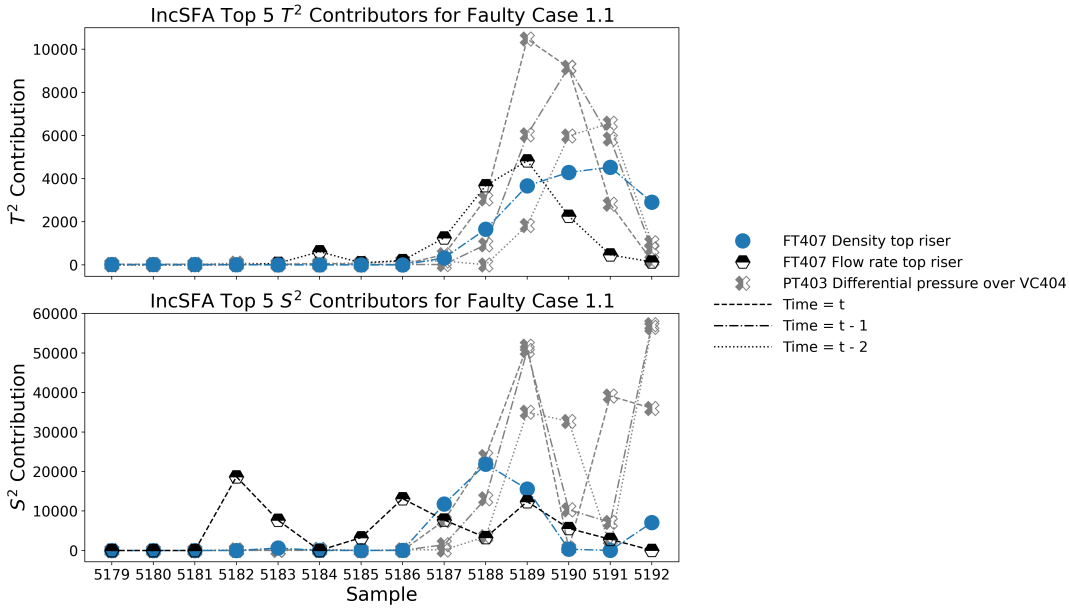

Figure 4.27: IncSFA fault diagnosis results for case 1 of fault 1

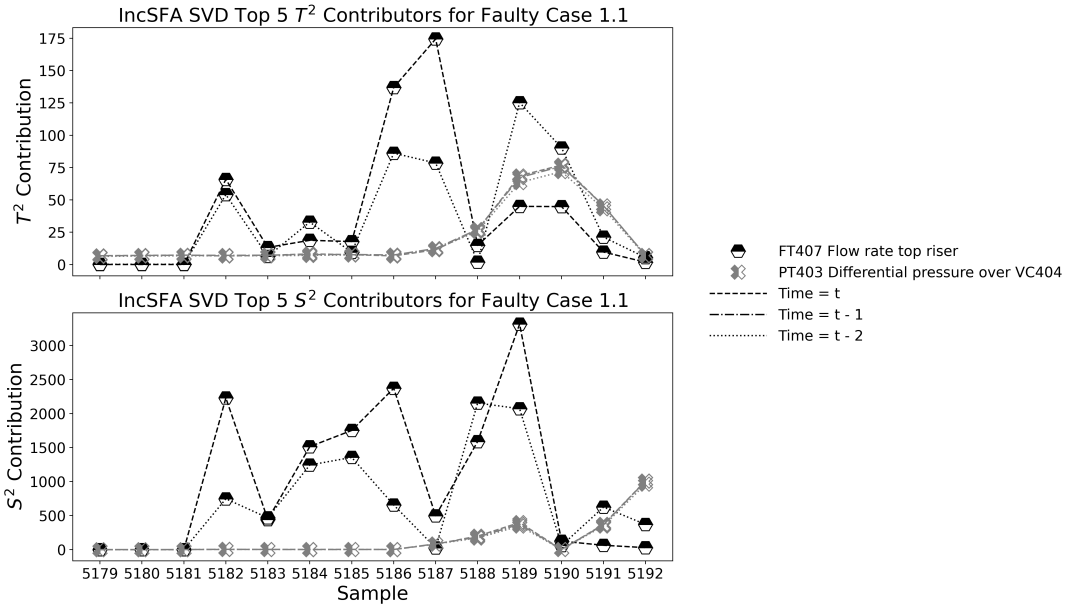

Figure 4.28: IncSFA SVD fault diagnosis results for case 1 of fault 1

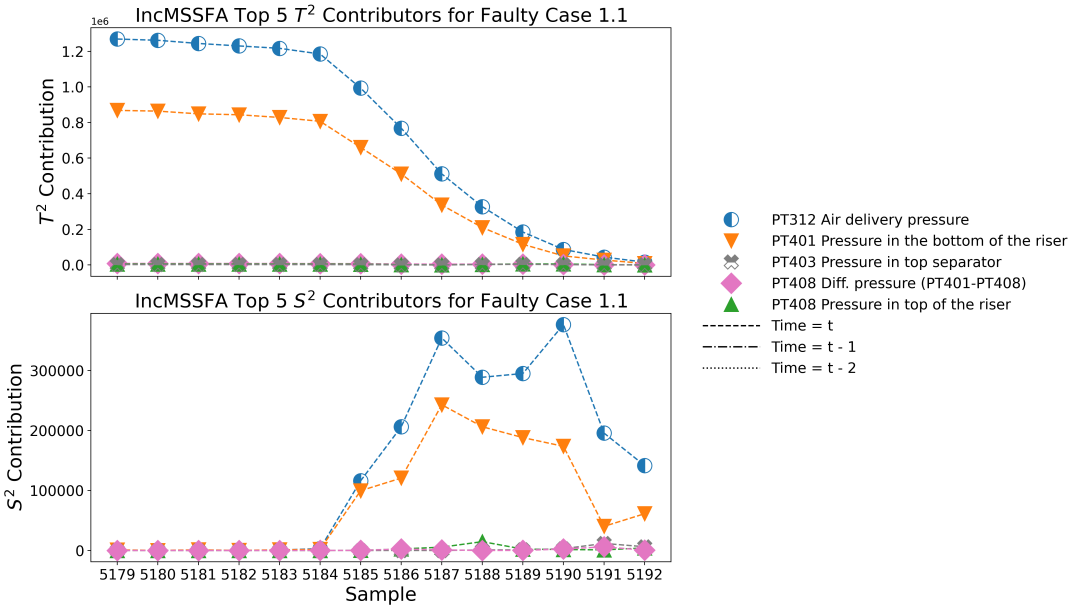

Figure 4.29: IncMSSFA fault diagnosis results for case 1 of fault 1

4.6 Conclusions

In this section, incremental model updates were explored. A monitoring framework for IncSFA was developed. In addition to this, an incremental extension of MSSFA termed IncMSSFA was introduced. These two algorithms were compared with another incremental SFA algorithm called RSFA. It was shown that IncSFA had the best complexity out of all the incremental algorithms investigated. The sparsity and interpretability investigation showcased the benefits of sparsity that were introduced with IncMSSFA. The investigation also showed the unstable nature of IncSFA. During the TEP and TPFF case studies, IncMSSFA achieved superior monitoring and diagnosis performance.

Chapter 5

Conclusions

In this thesis, MSSFA, a novel sparse SFA based algorithm was developed. SFA is well suited to process monitoring due to its explicit inclusion of process dynamics in its formulation. Sparse models are able to prevent overfitting and improve interpretability which are important for process monitoring. This new algorithm was created to improve upon SSFA which showed instability, poor monitoring performance, and low sparsity. The non-smooth objective function of sparse SFA and the SFA constraints were solved using APGD and manifold optimization. A fault detection and diagnosis framework was introduced to this algorithm. The TEP and TPFF data sets were then used to compare MSSFA with SFA, SSFA, and SPCA. MSSFA achieved improved interpretability and time complexity over the other algorithms, and was able to produce a more sparse model than SSFA. Case studies of the TEP and TPFF sets demonstrated the superior monitoring and diagnosis performance of MSSFA.

After this, SFA based incremental models were investigated. Incremental learning allows for adaptive models, removes the need to store historical data, and the need to feed in all training data at once which can be intractable for large data sets. To take advantage of the linear IncSFA complexity, a monitoring framework was developed for the algorithm. Another algorithm based on MSSFA and called IncMSSFA was introduced. IncMSSFA is the only incremental SFA based algorithm designed to produce sparse models. The performance of another incremental algorithm, RSFA, was compared to both IncSFA and IncMSSFA. With its linear complexity, IncSFA beat out the quadratic complexity of IncMSSFA and the cubic complexity of RSFA. As expected, IncMSSFA was the only model to achieve sparsity which also gave it superior interpretability. IncMSSFA also had the best monitoring and diagnosis performance of all the incremental algorithms for the TEP and TPFF case studies.

5.1 Recommendations

The next step for MSSFA/IncMSSFA is to include non-linearity in the model. The naive way to do this is to perform a nonlinear expansion on the input samples. Other ways to include this could be through the use of kernel or hierarchical methods. An investigation into these various methods is required to determine the best course of action.

Hierarchical methods have also been used to tackle high dimensional data sets; most commonly videos. The addition of sparsity into this domain may prove beneficial. The signals here correspond to pixels rather than physical values and so the interpretability aspect is not as important, but the prevention of overfitting is still relevant. In general, applying the models developed here to other application domains is worthy of investigation.

Another recommendation is to detail the convergence properties of the presented algorithms. This would require a more comprehensive education in the fields of topology and optimization, specifically the topics of manifolds and proximal algorithms.

Finally, it would also benefit the public to provide a more stream lined, efficient, and user-friendly implementation of these algorithms for practical use. One way to do this is to emulate NumPy and implement the algorithms in C for efficiency while providing an API to Python for ease of use.

References

[1] L. S. De Camp, *The ancient engineers*. Barnes & Noble Publishing, 1990.

[2] M. Xu, J. M. David, S. H. Kim, *et al.*, "The fourth industrial revolution: Opportunities and challenges," *International journal of financial research*, vol. 9, no. 2, pp. 90–95, 2018.

[3] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[4] V. E. Vinzi, W. W. Chin, J. Henseler, H. Wang, *et al.*, *Handbook of partial least squares*, vol. 201. Springer, 2010.

[5] P. Comon, "Independent component analysis," 1992.

[6] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 30, no. 1, pp. 179–196, 1995.

[7] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[8] B. R. Bakshi, "Multiscale pca with application to multivariate statistical process monitoring," *AIChE journal*, vol. 44, no. 7, pp. 1596–1610, 1998.

[9] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media, 2000.

[10] W. Li, H. H. Yue, S. Valle-Cervantes, and S. J. Qin, "Recursive pca for adaptive process monitoring," *Journal of process control*, vol. 10, no. 5, pp. 471–486, 2000.

[11] Y. Dong and S. J. Qin, "A novel dynamic pca algorithm for dynamic data modeling and process monitoring," *Journal of Process Control*, vol. 67, pp. 1–11, 2018.

[12] Q. Jiang and X. Yan, "Parallel pca–kpca for nonlinear process monitoring," *Control Engineering Practice*, vol. 80, pp. 17–25, 2018.

[13] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[14] J.-M. Lee, C. Yoo, and I.-B. Lee, "Statistical process monitoring with independent component analysis," *Journal of process control*, vol. 14, no. 5, pp. 467–485, 2004.

[15] Z. Ge and Z. Song, "Performance-driven ensemble learning ica model for improved non-gaussian process monitoring," *Chemometrics and Intelligent Laboratory Systems*, vol. 123, pp. 1–8, 2013.

[16] Y. Xu, S.-Q. Shen, Y.-L. He, and Q.-X. Zhu, "A novel hybrid method integrating ica-pca with relevant vector machine for multivariate process monitoring," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1780–1787, 2018.

[17] Z. Chen, S. X. Ding, K. Zhang, Z. Li, and Z. Hu, "Canonical correlation analysis-based fault detection methods with application to alumina evaporation process," *Control Engineering Practice*, vol. 46, pp. 51–58, 2016.

[18] Y. Liu, B. Liu, X. Zhao, and M. Xie, "A mixture of variational canonical correlation analysis for nonlinear and quality-relevant process monitoring," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6478–6486, 2017.

[19] Q. Zhu, Q. Liu, and S. J. Qin, "Concurrent quality and process monitoring with canonical correlation analysis," *Journal of Process Control*, vol. 60, pp. 95–103, 2017.

[20] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[21] V. R. Kompella, M. Luciw, and J. Schmidhuber, "Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams," *Neural Computation*, vol. 24, no. 11, pp. 2994–3024, 2012.

[22] C. Shang, F. Yang, B. Huang, and D. Huang, "Recursive slow feature analysis for adaptive monitoring of industrial processes," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8895–8905, 2018.

[23] C. Shang, X. Huang, F. Yang, and D. Huang, "Sparse slow feature analysis for enhanced control monitoring and fault isolation," in *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*, pp. 1–6, IEEE, 2019.

[24] C. Shang, F. Yang, X. Gao, X. Huang, J. A. Suykens, and D. Huang, "Concurrent monitoring of operating condition deviations and process dynamics anomalies with slow feature analysis," *AIChE Journal*, vol. 61, no. 11, pp. 3666–3682, 2015.

[25] H. Zhang, X. Tian, and X. Deng, "Batch process monitoring based on multiway global preserving kernel slow feature analysis," *Ieee Access*, vol. 5, pp. 2696–2710, 2017.

[26] H. Hotelling, "The generalization of student's ratio," in *Breakthroughs in statistics*, pp. 54–65, Springer, 1992.

[27] C. F. Alcala and S. J. Qin, "Analysis and generalization of fault diagnosis methods for process monitoring," *Journal of Process Control*, vol. 21, no. 3, pp. 322–330, 2011.

[28] P. J. Olver, C. Shakiban, and C. Shakiban, *Applied linear algebra*. Springer, 2018.

[29] A. M. Mood, *Introduction to the Theory of Statistics*. McGraw-hill, 1950.

[30] A. Kessy, A. Lewin, and K. Strimmer, "Optimal whitening and decorrelation," *The American Statistician*, vol. 72, no. 4, pp. 309–314, 2018.

[31] I. Rish and G. Grabarnik, *Sparse modeling: theory, algorithms, and applications*. CRC press, 2014.

[32] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[33] J. M. Lee, B. Chow, S.-C. Chu, D. Glickenstein, C. Guenther, J. Isenberg, T. Ivey, D. Knopf, P. Lu, F. Luo, *et al.*, "Manifolds and differential geometry," *Topology*, vol. 643, p. 658, 2009.

[34] P.-A. Absil, R. Mahony, and R. Sepulchre, "Optimization algorithms on matrix manifolds," in *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, 2009.

[35] H. Sato and K. Aihara, "Cholesky QR-based retraction on the generalized stiefel manifold," *Computational Optimization and Applications*, vol. 72, no. 2, pp. 293–308, 2019.

[36] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Computers & chemical engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[37] M. Grbovic, W. Li, P. Xu, A. K. Usadi, L. Song, and S. Vucetic, "Decentralized fault detection and diagnosis via sparse pca based decomposition and maximum entropy decision fusion," *Journal of Process Control*, vol. 22, no. 4, pp. 738–750, 2012.

[38] K. Liu, Z. Fei, B. Yue, J. Liang, and H. Lin, "Adaptive sparse principal component analysis for enhanced process monitoring and fault isolation," *Chemometrics and Intelligent Laboratory Systems*, vol. 146, pp. 426–436, 2015.

[39] X. Xiu, Y. Yang, L. Kong, and W. Liu, "Data-driven process monitoring using structured joint sparse canonical correlation analysis," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 361–365, 2020.

[40] P.-E. P. Odiowei and Y. Cao, "Nonlinear dynamic process monitoring using canonical variate analysis and kernel density estimations," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 36–45, 2009.

[41] C. Ruiz-Cárcel, Y. Cao, D. Mba, L. Lao, and R. Samuel, "Statistical process monitoring of a multiphase flow facility," *Control Engineering Practice*, vol. 42, pp. 74–88, 2015.

[42] S. Valle, W. Li, and S. J. Qin, "Selection of the number of principal components: the variance of the reconstruction error criterion with a comparison to other methods," *Industrial & Engineering Chemistry Research*, vol. 38, no. 11, pp. 4389–4401, 1999.

[43] E. L. Russell, L. H. Chiang, and R. D. Braatz, "Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 51, no. 1, pp. 81–93, 2000.

[44] R. Jenatton, G. Obozinski, and F. Bach, "Structured sparse principal component analysis," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 366–373, JMLR Workshop and Conference Proceedings, 2010.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[46] S. Gajjar, M. Kulahci, and A. Palazoglu, "Real-time fault detection and diagnosis using sparse principal component analysis," *Journal of Process Control*, vol. 67, pp. 112–128, 2018.

[47] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[48] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 1034–1040, 2003.

[49] E. Oja, "Principal components, minor components, and linear neural networks," *Neural networks*, vol. 5, no. 6, pp. 927–935, 1992.

[50] Y. Zhang, H. Zhou, S. J. Qin, and T. Chai, "Decentralized fault diagnosis of large-scale processes using multiblock kernel partial least squares," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 3–10, 2009.

[51] G. Li, S. J. Qin, and D. Zhou, "Geometric properties of partial least squares for process monitoring," *Automatica*, vol. 46, no. 1, pp. 204–210, 2010.

[52] Q. Jiang, X. Yan, H. Yi, and F. Gao, "Data-driven batch-end quality modeling and monitoring based on optimized sparse partial least squares," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 4098–4107, 2019.

[53] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

APPENDICES

Appendix A

Determining the Lipschitz Constant

A function is Lipschitz continuous with constant L if it satisfies

$$|g(x) - g(y)| \leq L|x - y| \tag{A.1}$$

for some norm and any values x and y.

For the purposes of the MSSFA algorithm, $g = \nabla \mathrm{Tr}(\mathbf{W}^\top \mathbf{B}\mathbf{W})$.

$$\nabla \mathrm{Tr}(\mathbf{W}^\top \mathbf{B}\mathbf{W}) = \mathbf{B}\mathbf{W} + \mathbf{B}^\top \mathbf{W} \tag{A.2}$$

Since covariance matrices are symmetric, $\mathbf{B}^\top = \mathbf{B}$.

$$g(\mathbf{W}) = 2\mathbf{B}\mathbf{W} \tag{A.3}$$

The left side of the Lipschitz condition is then

$$||2\mathbf{B}\mathbf{W}_1 - 2\mathbf{B}\mathbf{W}_2|| = 2||\mathbf{B}(\mathbf{W}_1 - \mathbf{W}_2)|| \tag{A.4}$$

after using the distributive property of matrix multiplication and taking the constant out of the norm. The following inequality holds when using a norm with the sub-multiplicative property such as the Frobenius norm.

$$2||\mathbf{B}(\mathbf{W}_1 - \mathbf{W}_2)||_F \leq 2||\mathbf{B}||_F||\mathbf{W}_1 - \mathbf{W}_2||_F \tag{A.5}$$

The right side of this inequality is the same as the right side of the Lipschitz condition with $L = 2||\mathbf{B}||_F$.

Appendix B

SSFA Instability

When the models were trained, data had to be preprocessed for the batch algorithms so that input signals had a mean of 0. For most of the algorithms, the variance of input signals had no significant effect on the performance of the model. This is because the models explicitly or implicitly include data whitening. However, SSFA showed numerical instability and provided unreliable results. To remedy this and provide a fair comparison, all the models had the input signals scaled to a variance of 1. The instability can be seen using the monitoring performance for IDV(4) of the TEP data set in Figure B.1. MSSFA monitoring results are also plotted here to show a comparison. The scale and general shape of the plots for normalized and unnormalized data are approximately equal for MSSFA while they drastically differ for SSFA.

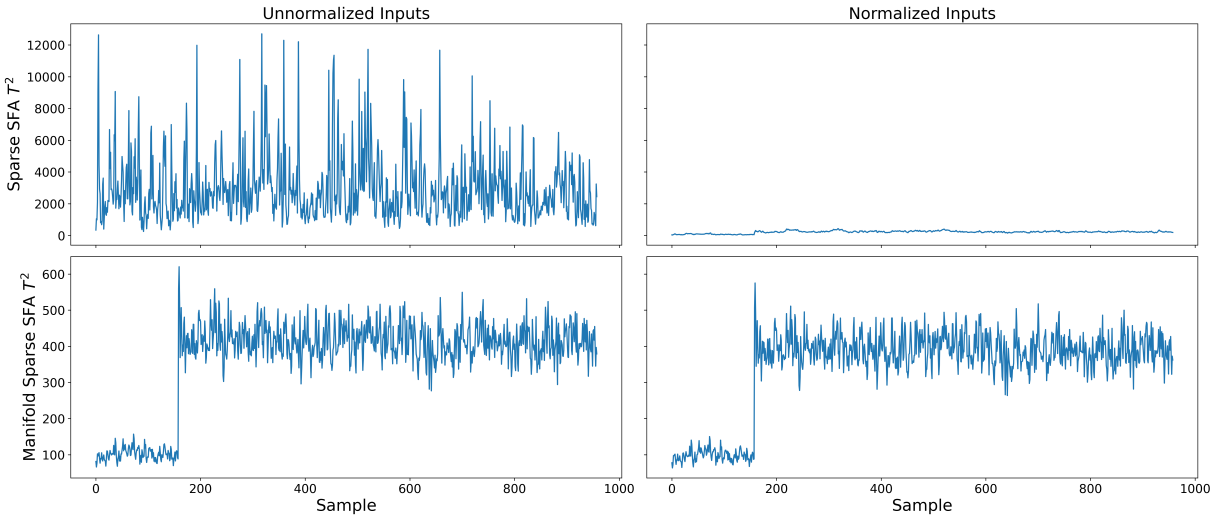

Figure B.1: Comparison of SSFA and MSSFA T^2 monitoring results for TEP IDV(4) when trained using normalized and unormalized data