Digital Chemical Engineering 8 (2023) 100116

Contents lists available at ScienceDirect

Digital Chemical Engineering

journal homepage: www.elsevier.com/locate/dche

Original article

# A one-class support vector machine for detecting valve stiction

Harrison O'Neill [a], Yousaf Khalid [b,*], Graham Spink [a], Patrick Thorpe [c]

[a] Department of Physical, Mathematical and Engineering Sciences, University of Chester, CH1 4BJ, United Kingdom
[b] Department of Engineering, Durham University, DH1 3LE, United Kingdom
[c] Spiro Control Ltd., Thornton Science Park, Chester, CH2 4NU, United Kingdom

ARTICLE INFO

*Keywords:*
Control loop performance monitoring
Control valve
Stiction
Principal component analysis
Linear discriminant analysis
Scikit-learn
Support vector machine
Tsfresh
Time series
Feature extraction

ABSTRACT

In industrial processes, control valve stiction is known to be one of the primary causes for poor control loop performance. Stiction introduces oscillatory behaviour in the process, leading to increased energy consumption, variations in product quality, shortened equipment lifespan and a reduction in overall plant profitability. Several detection algorithms using routine operating data have been developed over the last few decades. However, with the exception of a handful of recent publications, few attempts to apply classical supervised learning techniques have been published thus far. In this work, principal component analysis, linear discriminant analysis and a one-class support vector machine are trained to detect stiction using time series features as input. These features are extracted from the data using the `tsfresh` package for Python. The training data consists of simulated stiction examples generated using the XCH stiction model as well as other sources of oscillation. The classifier is subsequently benchmarked against closed-loop stiction data collected in an industrial setting, with performance exceeding that of existing methods.

## 1. Introduction

In a process plant, product quality, operational safety, and consumption of energy and materials are all linked to the performance of its control systems. As a result, the topic of control performance monitoring (CPM) has been an active area of research for the last three decades. A recent survey by Bauer et al. (2016) found that control engineers are often responsible for maintaining up to one thousand control loops at any given time. Such responsibility necessitates the use of automated systems and performance indices to assist in tracking down problematic loops. These kinds of tools are key in allowing control engineers to prioritise maintenance scheduling and avoid unnecessary shutdowns.

One of the main symptoms of a poorly controlled loop is oscillation. There are a number of causes of oscillatory behaviour, however the two most common are poor controller tuning and control valve stiction. Such behaviour in an industrial process can lead to increased energy consumption, variations in product quality, shortened equipment lifespan and a reduction in overall plant profitability. Stiction in particular is said to be the result of several factors: seal degradation, lubricant depletion, foreign debris and activation at metal sliding surfaces during high temperatures (Jelali and Huang, 2010). However, the primary contributor is thought to be the excessively tight packing surrounding the stem, due to the strict regulations regarding the leakage of volatile organic compounds into the body of the valve (see Fig. 1). This causes friction around the stem, restricting its movement and the responsiveness of the valve, ultimately leading to poor control.

The primary contribution of this work is the development and training of a new stiction classification procedure that combines principal component analysis, linear discriminant analysis and a one-class support vector machine. The method is shown to reliably detect stiction whilst requiring only the process variable (PV), the controller output (OP) and the control loop set point (SP). This system is trained using a detailed stiction simulation covering a wider variety of process parameters than previous works, and includes both integrating and self-regulating processes which are known to exhibit different stiction patterns. A simple procedure for selecting a suitable data window based on zero-crossings is given towards the end of the paper, and is shown to improve results when applied to industry data. When applied to the full set of industry benchmark data, the method produces the largest number of correct classifications (69/81), with an emphasis on high precision, meaning the method is unlikely to produce a false positive result. Moreover, when applied to a subset of the benchmark data containing only loops with known faults, the method ties for the highest number of correct classifications (22/26). As well as the ISDB data, the method is applied to an additional ten control loops with stiction to further demonstrate its effectiveness.

The remainder of this document is structured as follows. In Section 2, a brief introduction to the topic is provided as well as a survey

* Corresponding author.
*E-mail address:* yousaf.a.khalid@durham.ac.uk (Y. Khalid).

https://doi.org/10.1016/j.dche.2023.100116
Received 26 January 2023; Received in revised form 28 April 2023; Accepted 1 August 2023
Available online 10 August 2023

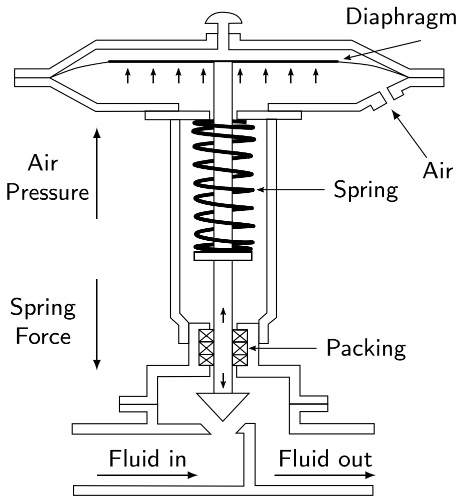



**Fig. 1.** Typical pneumatic control valve schematic.

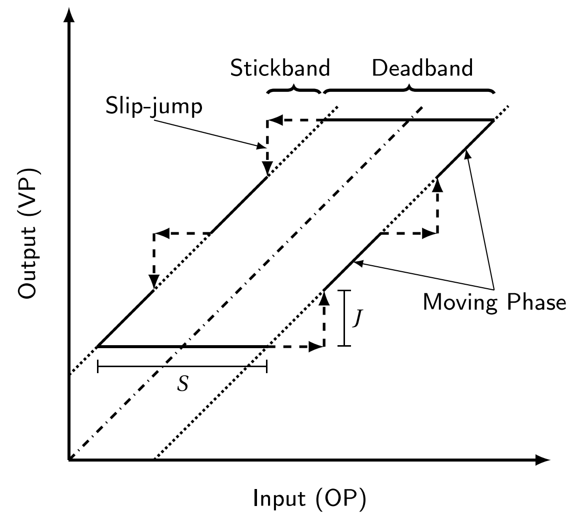



**Fig. 2.** Controller output vs valve position phase diagram.

of some of the methods developed for the automatic detection and quantification of control valve stiction. A description of the one-class support vector machine and dimensionality reduction techniques are also presented in this section. Section 3 describes the methodology of the proposed detection procedure, covering data generation, feature extraction and model training. In Section 4, a description of the metrics used to evaluate the model, as well as the results when classifying both simulated loops and real cases from industry. This section compares performance with a number of well-known methods on the widely used 'International Stiction DataBase' (ISDB) benchmark dataset presented in Jelali and Huang (2010) (available at Huang (2020), Bauer et al. (2020)). Analysis of incorrect loops and a method to address issues of data windowing are also provided. An additional ten control loops taken from the chemical industry and suffering from stiction are used for further testing. Finally, in Section 5 concluding remarks, suggested improvements and directions for further research are discussed.

## 2. Background

### 2.1. Valve stiction

The term 'stiction' is a portmanteau derived from the words static and friction, and is a phenomenon that has many definitions across the literature (Jelali and Huang, 2010). The one most generally accepted is that of Choudhury et al. (2005), which reads as follows:

> "The presence of stiction impairs proper valve movement, i.e. the valve stem may not move in response to the output signal from the controller or the valve positioner. The smooth movement of the valve in response to a varying input from the controller or valve positioner is preceded by a stickband and an abrupt jump termed the slip-jump. Its origin in a mechanical system is static friction, which exceeds the dynamic friction during smooth movement".

This definition is best explained by the behaviour of the controller output (OP) versus valve position (VP) plot seen in Fig. 2. For a perfectly healthy valve, the phase plot would travel along the dash-dotted line, where any amount of OP adjustment would result in an equal change in the position of the valve. In contrast, for a sticky valve, the static and kinetic/dynamic frictional components must be considered, leading to one of three states:

- **Sticking**. After stopping or changing direction, the valve position remains constant with the time, as it is unable to overcome the frictional forces.
- **Jumping**. The valve position changes abruptly, as the force supplied by the valve exceeds the friction.
- **Motion**. The valve moves freely until a reduction in velocity or change in direction. The valve is opposed only by dynamic friction forces.

The severity and widespread nature of the stiction phenomenon has led to many research articles and books over the last few decades, covering various sub-topics such as stiction modelling, detection, quantification, compensation and smart detection. In Jelali and Huang (2010), several stiction detection methods are presented and applied to a benchmark dataset of 93 control loops taken from industries such as chemicals, buildings, mining, metal processing, power generation and pulp & paper mills. The techniques featured in this book are the bicoherence and ellipse fitting method of Choudhury et al. (2004) (BIC), the cross-correlation method of Horch (1999) (CORR), the histogram method of Horch (2010) (HIST), the relay technique of Scali and Ghelardoni (2008) (RELAY), the area-peak method of Singhal and Salsbury (2005) (AREA), the curve fitting approach of He et al. (2007) (CURVE) and the two hammerstein identification methods of Lee et al. (2008), Karra and Karim (2009) (HAMM2, HAMM3). After applying each method to the available data, the take-home message was that no single method is able to produce consistent, reliable results in every situation. Many new detection systems are reviewed in Bacci di Capaci and Scali (2018), however due to a lack of benchmark results only the methods of Dambros et al. (2016) (SLOPE & ZONES) are discussed here.

One technique which has been published since Bacci di Capaci and Scali's review is the non-linear principal component analysis application of Teh et al. (2018) (NLPCA-AC). Although principal component analysis is closely related to machine learning, as there is no prior training process this method is considered separate from the methods discussed in Section 2.2. The new NLPCA-AC method attempts to improve on previous work by Zabiri and Ramasamy (2009), which had also used NLPCA, through additional pre-processing, post-processing and the consideration of the number of zero-crossings in the auto-covariance function, a metric which is also used in Thornhill et al. (2003). NLPCA-AC has been applied to the benchmark dataset and exceeds the total number of correct responses previously achieved by the BIC method. This is recognised in the paper and is used to support the claim that the method is more reliable. Whilst the results are impressive, caution is advised when using the number of correct verdicts as a means of determining the best overall method, particularly in the case of BIC which is predominantly limited by the window length of the test data. In an online setting, the length of the data should only be an issue for loops with frequent step changes (BIC requires a minimum of 1000 samples without an abrupt set point change). Recent advances surrounding the bicoherence statistic indicates that this limitation can be eased further, as (Lang et al., 2018) show an improved 'bihocerence' statistic for shorter time series that is successfully applied to CHEM_22, a loop containing only 721 samples.

In contrast to the recent wave of machine learning-based techniques presented in recent years (discussed in the following section), Damarla et al. (2022) instead propose a sigmoid function-based method for the detection of stiction in control valves. Applicable to both stationary and non-stationary signals, this method can be used across various control loops, including flow, temperature, concentration, pressure, and analyzer-related loops. The proposed method asserts that if oscillations are caused by stiction, the OP($k$) vs $\Delta$PV($k$) phase plot will display an s-like shape or sigmoid-like curve, which is a distinctive characteristic of a sticky control valve. By fitting the sigmoid function to the data, a correlation coefficient ($R$) can be computed. If $R$ is greater than or equal to a chosen threshold ($R_{\text{threshold}}$), stiction is deemed the cause of oscillations. This method was tested on twenty control loops from the ISDB database, as well as two flow control loops from an oil sands industry, demonstrating its effectiveness beyond the ISDB dataset.

In Zheng et al. (2021), the authors propose a stiction detection method that takes advantage of the slow PV response relative to OP changes via K-means clustering. Although technically a machine learning method, what differentiates it from those in the following section is that it is unsupervised i.e., no labels are required during training. This method is compatible with both stationary and nonstationary signals of varying lengths. Applicable to flow, temperature, pressure, and concentration loops, it can identify various faulty valves, including frozen, clogged, and severely stuck valves. Although the method is robust against low levels of noise in PV or OP signals, moderate to high noise levels necessitate denoising before analysis. By examining the distribution of the ratio of $\Delta$OP to $\Delta$PV, the method can determine the likelihood of valve stiction and enhance stiction detection reliability using a moving window-based approach. Just like the sigmoid method, this method was tested on twenty control loops from the ISDB database, as well as three loops from the oil sands industry.

### 2.2. Supervised learning-based methods

The methods discussed in this section are those which are able to identify, or quantify stiction through supervised machine learning. This process is where labelled training data are fed to a classification/regression algorithm, enabling it to make predictions for new data based on patterns learned during training. A classifier is an algorithm which is designed to predict discrete class labels e.g. 'stiction', 'aggressive tuning' or 'disturbance', whereas a regressor instead learns to predict continuous values. There are four stiction detection/quantification methods described in the literature that match this description, the first of which, developed by Farenzena and Trierweiler (2009), is a neural network trained using simulated data to quantify stiction through prediction of the parameters $S$ (deadband + stickband) and $J$ (slip-jump). The simulation of Farenzena and Trierweiler consists of control loops with varying characteristics such as controller parameters $K_p$ and $T_i$, process parameters $\tau$ (time constant) and white noise variance, and of course the stiction model parameters $S$ and $J$. The following seven values are considered as inputs for the neural-network-based quantification scheme:

- $\Delta PV$: The difference between the maximum and minimum value in the $PV$ time series;
- $\Delta OP$: The difference between the maximum and minimum value in the $OP$ time series;
- $\Delta PV/\Delta OP$: The ratio between $\Delta PV$ and $\Delta OP$;
- $E_w$: The integral of the squared error of the difference between $PV$ and the best wave-shape curve interpolation ($PV_w$) (Rossi and Scali, 2005);
- $E_T$: The integral of the squared error of the difference between $PV$ and the best triangle-shaped curve interpolation ($PV_T$);
- $ZC$: The number of zero-crossings in the mean corrected data;
- $ZC_{ACV}$: The number of zero-crossings in the autocovariance function (Thornhill et al., 2003).

However, after applying feature selection based on stepwise regression, only the $\Delta PV$, $\Delta PV/\Delta OP$ and $ZC_{ACV}$ features were found to be useful. This method is unfortunately limited to just quantification, but the proposition of combining machine learning with simulated training data has served as inspiration for the remaining methods in this section, as well as the work presented in this paper.

One method developed more recently is the stiction detection network (SDN) by Amiruddin et al. (2019), who have designed a learning-based strategy which is once again powered by a neural network. The SDN is trained using simulated cases of stiction, but also other sources of oscillation such as poor tuning and external disturbance. This is important if a classifier is going to be applied to a real control loop, as one trained using only stiction cases runs the risk of being unable to distinguish stiction from a regular sinusoidal oscillation. Where the SDN differs significantly from the previous approach, is that rather than selecting a handful of calculated features as input, the authors consider 500 data points from both the OP and PV data, and compute a new variable

$$D_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}, \tag{1}$$

where $x_i, y_i$ are the $i$th elements of the OP and PV data, and $x_c, y_c$ are their respective means. The output is then interpreted as a probability that the loop is suffering from stiction, making the SDN a binary classifier, as opposited to a multi-class classifier which would attempt to identify the other cases also.

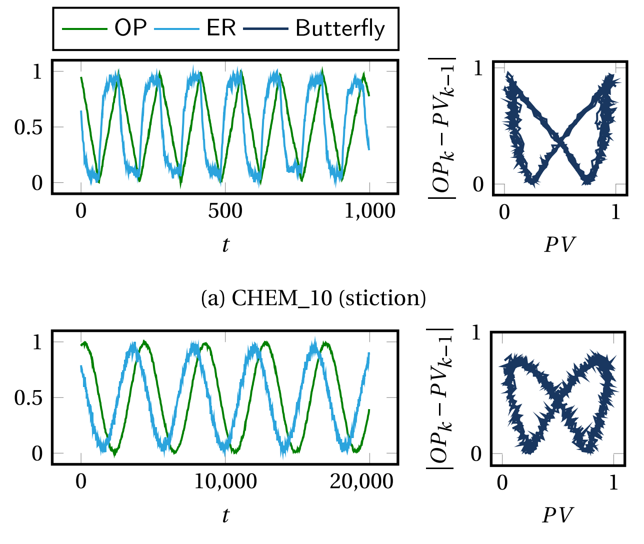
The SDN is one of the highest performing detection methods reviewed here. Although others may report higher accuracy, this might be a result of limiting their application to a subset of the available loops in the ISDB. BIC, for example, is applied to just 57 loops, eliminating many of the shorter length, manually controlled and heavily quantised loops. In applying SDN to the same 57 loops tested by BIC, SDN achieves 50/56[1] correct responses, compared with 48/57 from BIC. This is a surprising result given that the network is trained using just a single process model.

Shortly after publishing SDN, the same group produced a second method with the name "butterfly-shape detection" (BSD) (Kamaruddin et al., 2020). The algorithm for BSD is derived from Stenman's one-parameter stiction model:

$$x_k = \begin{cases} x_{k-1} & \text{if } |u_k - x_{k-1}| < d, \\ u_k & \text{otherwise} \end{cases} \tag{2}$$

where $u$ and $x$ are the valve input (OP) and output (VP), respectively, and $d$ is the band of valve stiction. Based on this model, if the valve is stuck then the output remains at its previous value. Taking this as the starting point, and using PV instead of VP in place of $x$, the graph of the $|OP_k - PV_{k-1}|$ was plotted against PV and a butterfly-like pattern was observed for stiction cases (see Fig. 3(a)). Note that the reason for replacing VP with PV is that the adoption of smart valves is still relatively low, meaning that the valve position is rarely available. Confirmation of stiction from the butterfly plot still requires manual inspection, so to automate this the authors introduce the Identification of Round Objects Method (IROM) to count the number of enclosed areas of the plot. The background of the plot counts as one, and so should each wing of the butterfly, so an IROM index > 1 is considered as sufficient evidence of stiction. The learning element of the method comes once stiction has been confirmed, as a pre-trained neural network is used later to quantify the level of stiction. The network is a

[1] Verdict missing for CHEM_41, which is experiencing OP saturation.




(a) CHEM_10 (stiction)

(b) Sinusoidal disturbance (no stiction)

**Fig. 3.** Butterfly plots for a real stiction case and a simulated non-stiction case. Along with controller output (OP) and error ($ER = SP - PV$) time plot.

convolutional neural network (CNN) which is a powerful technique used in image classification (Rawat and Wang, 2017). Rather than predicting the values of $S$ and $J$, the stiction class is divided into three sub-classes: weak, medium and strong stiction, and the CNN is used to classify loops accordingly.

The BSD algorithm performs well on a selection of industrial control loops, however the IROM index approach for identifying stiction can lead to false positives. As evidenced by Fig. 3(b), the butterfly pattern, and therefore the number of enclosed areas indicated by the IROM index, are practically identical for sinusoidal disturbances, which could be a problem depending on how often such disturbances occur in real processes. This issue could be eliminated by referring to a CNN for both detection and quantification, as such a technique should be capable of spotting the subtle differences between plots.

Another recent stiction detection and quantification method is that of Henry et al. (2020), which makes use of the data simulation process described in Amiruddin et al. (2019), Kamaruddin et al. (2020). Here, the raw time series data (PV-OP) is transformed into an image using the unthresholded recurrence plot (URP). The authors make use of AlexNet, a well-known pre-trained network that was the first convolutional neural network to win the ImageNet Large Scale Visual Recognition Challenge in 2012 (Krizhevsky et al., 2012). This network is partially re-trained by securing the weights contained in the first few layers and allowing only those in final layers to be modified. In essence, this trains AlexNet to be specialised in recognising the URP simulated control loop data. The 4096 nodes contained in the second fully connected layer of the network are used as features and are fed into a PCA model for stiction detection. Both Hotelling's $T^2$ and the $Q$-statistic are used to identify the stiction, however the $Q$-statistic was found to be better at distinguishing different levels of stiction. A support vector machine with linear kernel is used to classify stiction as either weak (needs to be monitored) or strong (needs to be fixed) stiction. The method is applied to 78 loops from the industry benchmark and achieves 55 correct verdicts. The ability to track stiction progression over time using the Q-statistic as well as the output of the SVM makes this method very compelling.

Finally, a recent method is the convolution based approach for industrial time-series data presented in Zhang et al. (2022), one which utilizes self-supervised contrastive learning and specifically targets valve stiction. The approach comprises two components: data transformation and representation learning. A temporal distance matrix transformation method is introduced, converting raw time-series data into temporal distance matrices based on dynamic time warping (DTW) distances, effectively capturing temporal and spatial information. A convolution-based encoder is employed to encode the matrices into embedding representations, guided by a new multitimescale feature consistent constraint (MTFCC) for self-supervised representation learning. Finally, a general fault detection framework consisting of an unsupervised feature learning module and a detection module is presented, and its effectiveness in valve stiction detection tasks is demonstrated using industrial benchmark datasets, a hardware experimental system, and real industrial environments (see Table 1).

### 2.3. One-class support vector machines

The one-class support vector machine (OCSVM) is an extension of standard multi-class SVM of Cortes and Vapnik (1995). The authors, Schölkopf et al. (2000), believed that the general multi-class approach was poorly suited for novelty detection and so developed an alternative solution for the one-class case. The main difference is that the original SVM requires examples of both 'normal' and 'faulty' data to train the model, whereas the one-class SVM requires only the former. The usual way of applying the one-class SVM is to train using normal operating data such that an issue is detected if the process deviates significantly from the expected behaviour, requiring further investigation to obtain a specific diagnosis. Here instead, the one-class SVM is trained using stiction as the 'normal' case, this way a stiction/no stiction diagnosis is given immediately. A mathematical description of the one-class SVM is provided below.

Consider the training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m\}$, where each $\boldsymbol{x}_i$ is a feature vector that lies in some feature-space $X$. Let $\phi$ be a function that maps $X \rightarrow F$, where $F$ is a higher dimensional feature space, and $k$ a kernel function such that

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi\left(\boldsymbol{x}_i\right) \cdot \phi\left(\boldsymbol{x}_j\right). \tag{3}$$

For data which is not linearly separable in $X$, a kernel function that is used often is the radial basis function (RBF)

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right). \tag{4}$$

The parameter $\gamma$ controls how the training data influences the shape of the decision boundary; high values mean that data closer to the decision boundary have greater influence, whereas lower values increase the weight given to points further away.

The goal of the One-Class SVM is to create a function $f$ which returns '+1' for all points within the region containing the training data and '−1' everywhere else. This is achieved by mapping the data into the new feature-space $F$ corresponding to the chosen kernel, and finding a hyperplane that separates the data from the origin with maximal margin. New data are then classified based on where they are positioned in relation to the decision boundary when viewed in original-space $X$ (see Fig. 4). This function can be obtained through the following optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{w},\xi,\rho} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \\ \text{s.t.} \quad & \boldsymbol{w}\cdot\phi(\boldsymbol{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0. \end{aligned} \tag{5}$$

Here, $\xi_i$ are slack variables, $\rho$ is the bias term and $\nu \in (0, 1]$ is a parameter which controls what portion of the supplied training data can be regarded as outliers. Low values for $\nu$ can resulting in overfitting (Fig. 4(a)), so it is advisable to tune this hyperparameter in order to obtain a more general decision boundary (Fig. 4(b)). Note that the issue of over/under-fitting appears frequently in machine learning applications and is commonly referred to as the bias–variance tradeoff.

The desired decision function is of the form

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho\right), \tag{6}$$

**Table 1**
A selection of stiction detection methods that have been tested on a significant portion of the 81 loops from benchmark data of Jelali and Huang (2010), based on and updated from (Bacci di Capaci and Scali, 2018).

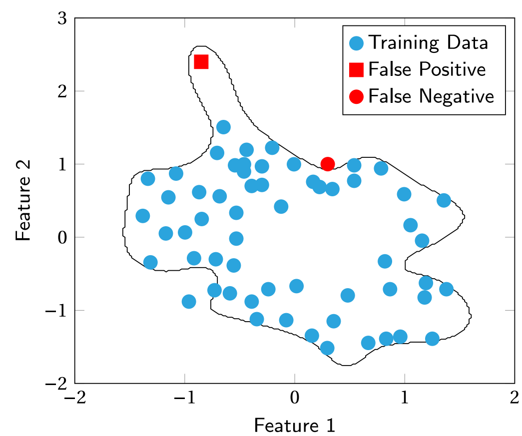
| Method | | Features | | |
|---|---|---|---|---|
| Name | Author(s) | Type | Loop applicability | Correct verdicts |
| CORR | Horch (1999) | Cross-correlation | No LC | 20/49 |
| HIST | Horch (2010) | Statistics | All type | 31/65 |
| BIC | Choudhury et al. (2004) | NL detection | All | 48/57 |
| RELAY | Rossi and Scali (2005) | Waveform shape | All | 34/69 |
| AREA | Singhal and Salsbury (2005) | Waveform shape | No LC | 30/48 |
| CURVE | He et al. (2007) | Waveform shape | All | 27/63 |
| HAMM2 | Lee et al. (2008) | Hammerstein Identification | All | 40/76 |
| HAMM3 | Karra and Karim (2009) | Hammerstein Identification | All | 44/81 |
| SLOPE | Dambros et al. (2016) | Waveform shape | All | 29/61 |
| ZONES | Dambros et al. (2016) | Waveform shape | All | 28/61 |
| NLPCA | Teh et al. (2018) | Nonlinear PCA | All | 54/81 |
| SIGMOID | Damarla et al. (2022) | Shape-based | All | 17/20 |
| KMEANS | Zheng et al. (2021) | K-Means (w/ Quantification) | All | 16/20 |
| SDN | Amiruddin et al. (2019) | ML-based (Neural Network) | All | 61/80 |
| BSD | Kamaruddin et al. (2020) | ML-based (Convolutional Neural Network) | All | 20/26 |
| CNN-PCA | Henry et al. (2020) | ML-based (Convolutional Neural Network) | All | 55/78 |
| MTFCC | Zhang et al. (2022) | ML-based (SVM) | All | 22/26 |

where coefficients $\alpha_i$ are obtained via the solution of the dual problem

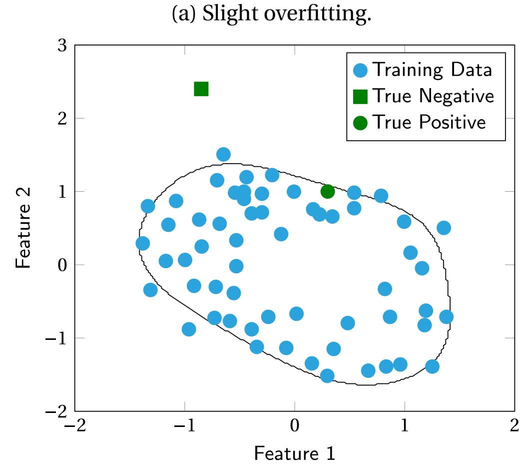
$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{ij}\alpha_i\alpha_j k\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right) \\ &\text{s.t.} \quad 0 \le \alpha_i \le \frac{1}{\nu N}, \quad \sum_i \alpha_i = 1. \end{aligned} \tag{7}$$

This can be solved using standard quadratic programming routines. The training data with non-zero $\alpha_i$ are called support vectors, which is where the SVM gets its name.

For those interested in reading about the one-class SVM applied specifically to a control-related problem, see Mahadevan and Shah (2009), where the method is applied to a Tennessee Eastman process simulation to detect 18 out of 21 faults. The one-class SVM is compared with principal component analysis (PCA) and dynamic principal component analysis (DPCA) and outperforms both with respect to detection speed and accuracy for most faults.




(a) Slight overfitting.




(b) Balanced fit.

**Fig. 4.** Illustrative example of the one-class support vector machine. In (a), the two new examples (○ = inlier, □ = outlier) are falsely classified due to the overfitted decision boundary. In (b), the new examples are correctly classified as a result of the simpler decision boundary.

### 2.4. Techniques for dimensionality reduction

For learning problems involving many features, a good solution can be difficult to obtain even if said features are useful for prediction. The higher the dimensionality of the data, the more training samples are required to obtain a good result. This problem is referred to as the curse of dimensionality. Many reduction techniques have been developed to tackle this issue, two of which are discussed in this section and applied as part of the proposed stiction detection method in Section 3.

#### 2.4.1. Principal component analysis

Principal component analysis, or PCA, is an unsupervised technique with a variety of applications including dimensionality reduction, data compression, feature extraction and data visualisation. PCA can be defined as the orthogonal projection of data onto a lower dimensional linear space, such that the variance of the projected data is maximised (Hotelling, 1933). Following this definition, the method is applied as follows.

First, let $X \in \mathbb{R}^{m\times n}$ be a feature matrix containing $m$ observations and $n$ features, then compute the $n \times n$ covariance matrix

$$S_t = \frac{1}{m}(X-M)^T(X-M), \tag{8}$$

where $M$ is an $m\times n$ matrix whose columns are populated by the means of each of the features in $X$.

Next, find all pairs of eigenvectors and eigenvalues of $S_t$, sorting each pair by eigenvalue in descending order. Define a matrix $W$ with columns containing the eigenvectors corresponding to the ordered eigenvalues i.e. let the first column contain the eigenvector with the largest eigenvalue, and so on. The original data $X$ is mapped onto a new coordinate system by multiplication with $W$, where the axes of this new frame of reference are called the principal components.

$$X_{\text{PCA}} = XW. \tag{9}$$

A reduction in dimensionality is achieved by selecting a subset of the eigenvectors to place in $W$. The variance explained by each principal component is determined by its associated eigenvalue, and thus a method for choosing the number of components to use is to select as many as are required as to retain the desired amount of variance. This is typically defined as a percentage of the total variance.

#### 2.4.2. Linear discriminant analysis

Linear discriminant analysis, or LDA, is similar to PCA in that it projects the original data into lower dimensions. Whilst PCA creates components that maximise variance, LDA selects components that maximise the distance between labelled classes. This means that unlike PCA, it is a supervised method that requires labelled data to perform the transformation. LDA is also restricted by the number of classes, as the maximum number of dimensions that can be obtained from the method is $c-1$, where $c$ is the number of unique class labels.

The method is described as follows. As before, let $X \in \mathbb{R}^{m\times n}$ be a feature matrix containing $m$ observations and $n$ features. Only now suppose that each observation in $X$ can be categorised into one of $c$ classes, with a class label stored in the vector $\boldsymbol{y} \in \mathbb{R}^m$. The 'within-class' covariance matrix is then computed by separating $X$ into groups corresponding the class labels in $\boldsymbol{y}$, then summing the individual covariance matrices multiplied by the class priors

$$S_w = \sum_{i=1}^{c} \frac{p_i}{m_i}(X_i - M_i)^T(X_i - M_i). \tag{10}$$

Here $X_i$ are the sub-matrices of $X$ corresponding to class $i$, $M_i$ are the matrices containing the means of each feature column in $X_i$, $m_i$ are the number of observations of class $i$ and $p_i = m_i/m$ are the class priors estimated as the proportion of observations of class $i$ with respect to the total. The 'total' covariance matrix is computed just as in PCA,

$$S_t = \frac{1}{m}(X - M)^T(X - M), \tag{11}$$

where $M$ is a matrix whose columns are populated by the means of each feature in $X$. The 'between-class' covariance matrix is result of subtracting the 'within-class' covariance from the 'total' covariance

$$S_b = S_t - S_w. \tag{12}$$

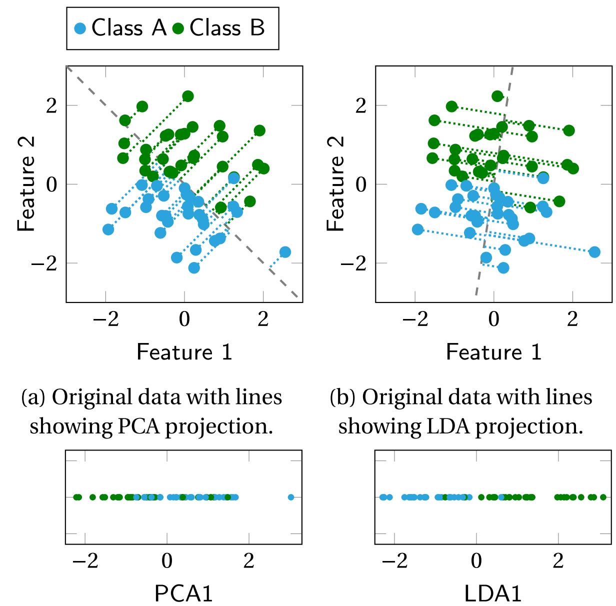
The original $n$-dimensional data is projected down to $(c-1)$ dimensions via the $n \times (c-1)$ matrix $W$ whose columns are formed by the eigenvector solutions of the generalised eigenvalue problem

$$S_b \boldsymbol{w}_i = \lambda_i S_w \boldsymbol{w}_i. \tag{13}$$

Just as with PCA, the eigenvectors are ordered by their corresponding eigenvalues. The projected data is obtained via the multiplication

$$X_{\text{LDA}} = XW. \tag{14}$$

In most applications, the resulting transformation should preserve the class clusters despite the reduction in dimensionality. An example of both PCA and LDA being applied to a dummy 2-D dataset is presented in Fig. 5. The top two graphs show how the original data is projected into a single dimension, and the bottom two show where the points lie on the new axis. In Fig. 5(c), the classes overlap heavily, meaning that any classifier applied to this data will not perform well. On the otherhand, in Fig. 5(d), there is minimal overlap and the subsequent classification problem is now much easier.




(a) Original data with lines showing PCA projection.

(b) Original data with lines showing LDA projection.

(c) PCA representation of data in one dimension.

(d) LDA representation of data in one dimension.

**Fig. 5.** A comparison of PCA and LDA used to reduce dummy two-dimensional data to a single dimension.

## 3. Methodology

In this section, the procedures for generating data, performing feature extraction and training the one-class support vector machine are described. Section 3.1 provides the details of the MATLAB & Simulink setup used to generate the training data. Section 3.2 describes the feature extraction and dimensionality reduction techniques used to provide inputs for the classifier. Section 3.3 defines the classifier training procedure.

### 3.1. Generating training data

The data used for training the classification model is generated entirely via simulation in MATLAB & Simulink. A simplified view of the Simulink model can be seen in Fig. 6, the elements of which are detailed throughout this section. In contrast with previous methods, two process models are used for generating the training data: a self-regulating process and an integrating process (Figs. 7 and 8). This is important, as stiction is known to present differently for integrating processes, as demonstrated in Figs. 9(a) and 9(b). Improving on the simulation work of Amiruddin et al. (2019), many of the process parameters used to generate the training data are altered with each simulation. This is in response to the findings of Rossi and Scali (2005), where it is demonstrated that parameters such as time constant $\tau$ and time delay $T_d$ have a noticeable influence on the shape of the PV and OP time series. This is evidenced in Fig. 10, where adjusting the ratio between $T_d$ and $\tau$ results in significantly different stiction patterns.

The parameters considered and their respective ranges are defined in Table 2. To avoid simulating all possible combinations, a Monte Carlo style approach is adopted where parameters are selected randomly with each simulation.

#### 3.1.1. Cases considered

In accordance with the scenarios presented in Choudhury et al. (2004), four cases are considered for simulation:

- A well-tuned loop with no oscillation;

**Table 2**
Range of parameters used for the simulations.

| Cases All | Parameter | Description | Range |
|---|---|---|---|
| | $K$ | Process gain | $[1 : 0.1 : 3]$ |
| | $\tau$ | Process time constant | $[[0.1 : 0.1 : 1.9][2 : 1 : 11][12 : 6 : 54][60 : 12 : 240]]$[a] |
| | $T_d$ | Process dead time | $[[0.1 : 0.1 : 0.9][1 : 1 : 9][10 : 5 : 60]]$ |
| | $T_s$ | Sampling period | $[0.1, 1, 3, 5, 10, 12, 15, 20, 30, 60]$ |
| | $N$ | Sample length | $[500 : 250 : 2000]$ |
| | $SP$ | Setpoint | $[30 : 1 : 70]$ |
| | $V$ | Variance of noise | $[0.0001 : 0.001 : 0.005] \times SP$ |
| | $K_p$ | Proportional gain | Determined by `pidtune` function |
| | $K_i$ | Integral gain | Determined by `pidtune` function |
| Stiction | | | |
| | $S$ | Deadband + stickband | $[1 : 0.1 : 10]$ |
| | $J$ | Slip-jump | $[1 : 0.1 : 10]$ |
| Disturbance | | | |
| | $A$ | Sine wave amplitude | $[0.25 : 2]$ |
| | $\phi$ | Sine wave phase | $[0 : 2\pi]$ |
| | $f$ | Sine wave frequency | $[0.005 : 0.005 : 0.05] \times 1/T_s$ |
| Aggressively Tuned | | | |
| | $K_p$ | Proportional gain | $0.75K_u$ |

[a]Time constant $\tau$ is restricted to $[0.1 : 0.1 : 1.9]$ for integrating cases to represent fast flow dynamics.

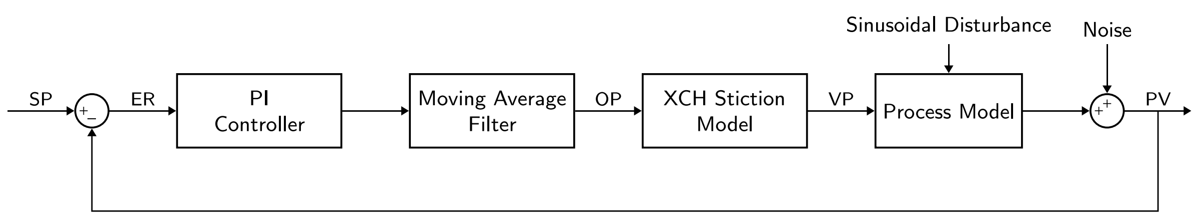



**Fig. 6.** A simplified view of the Simulink model used to generate the control loop data.

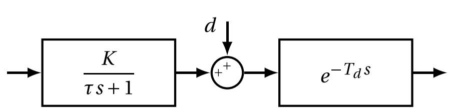



**Fig. 7.** Self-regulating process based on Kano et al. (2004).

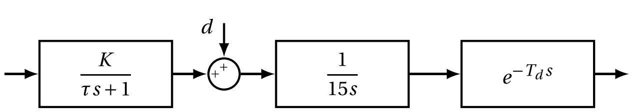



**Fig. 8.** Integrating process based on Kano et al. (2004).

- An aggressively tuned, oscillating loop;
- A well-tuned loop with sinusoidal disturbance;
- A well-tuned loop suffering from stiction.

Although the non-stiction examples are not technically required for training the one-class SVM, they are however necessary for the dimensionality reduction, specifically linear discriminant analysis. This will be discussed in greater detail in Section 3.2. Non-stiction cases are also useful in validation and testing stage.

#### *3.1.2. Controller tuning*

In all cases a PI controller is used and suitable values for the controller gains $K_p$ and $K_i$ must be automatically determined. Although there is autotuning functionality built into the PID Controller block in Simulink, at the time of writing it does not seem possible to automate this from within a MATLAB script. To address this, the process model and moving average filter are defined in MATLAB, allowing the use of the `pidtune` function which automatically determines the optimal controller gains. The parameters are then provided to the Simulink model using the `set_param` function.

For the aggressive tuning case, the proportional gain $K_p$ is set to 3/4 of the ultimate gain $K_u$ (calculated by applying the `margin` function to the MATLAB representation of the open-loop system, without stiction). For perspective, the commonly used Ziegler–Nichols tuning rules suggest a value of $0.45K_u$ as a starting point for further manual tuning. An example of a simulated aggressively tuned case can be seen in Fig. 9(c), exhibiting the expected rapid and erratic oscillations.

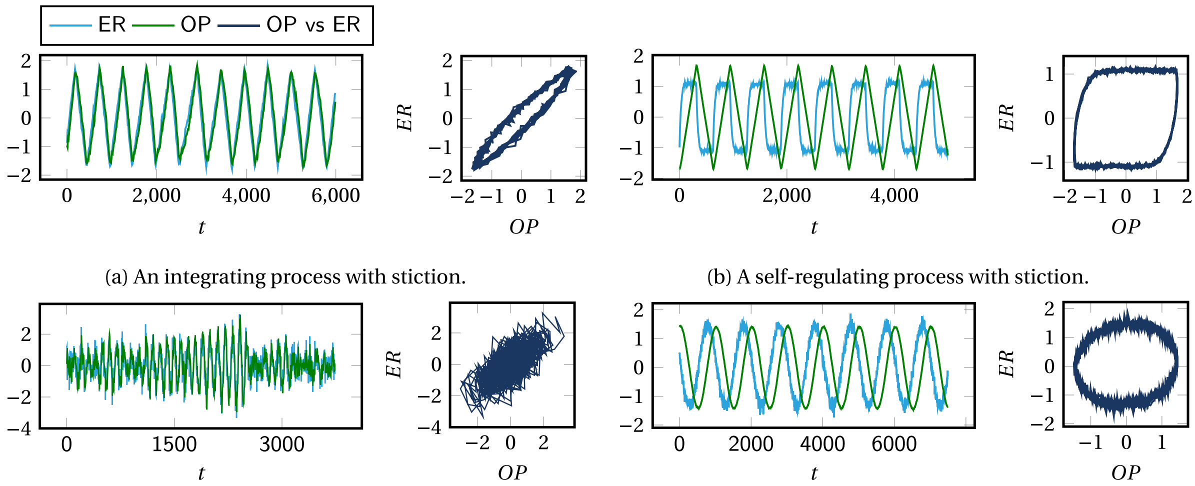
#### *3.1.3. Noise and disturbance*

For the externally disturbed cases, a sine wave of fixed amplitude and frequency is added to the system at the location marked by $d$ in Figs. 7 and 8. With each simulation, the amplitude and phase are selected randomly in the ranges $[0.25 : 2]$, $[0 : 2\pi]$ respectively. The frequency is dependent upon the sampling period $T_s$ (measured in seconds) and is a random value in the range $1/T_s \times [0.005 : 0.005 : 0.5]$. This produces sinusoidal oscillations with frequencies similar to the stiction cases, which should be difficult for the classifier to distinguish. The hope is that this will force LDA to find projections that differentiate stiction from regular sinusoidal oscillations. An example of a loop disturbed by a sine wave can be seen in Fig. 9(d). For all other cases the disturbance is switched off by commenting out the sine wave block.

Gaussian distributed white noise with zero mean and constant variance is added to the process value ($PV$) to simulate sensor noise. The variance of the noise is proportional to the setpoint, and is randomly selected from the range $[0.0001 : 0.001 : 0.005] \times SP$. As the minimum and maximum setpoint are 30 and 70 respectively, this means that the minimum and maximum possible variance are 0.003 and 0.2870, which is similar to that used in Amiruddin et al. (2019).

#### *3.1.4. Setpoint*

The stiction patterns in both the OP and ER variables are best preserved in the absence of abrupt changes in control loop setpoint. For this reason, the preferred setpoint mode for each simulation is a fixed value, where the transient portion of the data found at the




(a) An integrating process with stiction.

(b) A self-regulating process with stiction.

(c) A self-regulating process with aggressive tuning.

(d) A self-regulating process with sinusoidal disturbance.

**Fig. 9.** Simulated examples of oscillation in a control loop.

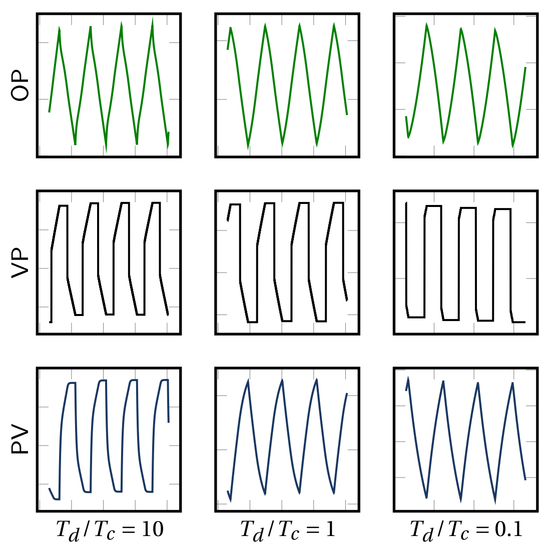



**Fig. 10.** The effect of varying the ratio between time delay $T_d$ and time constant $T_c$ (Rossi and Scali, 2005).

beginning of each simulation is discarded. Given that the data will be standardised prior to feature extraction, it is unlikely the choice of setpoint contributes anything to the stiction behaviour, that said, the value is randomly chosen from the range $[30 : 1 : 70]$ as a precaution.

An observation from the benchmark data of Jelali and Huang (2010) is that due to the cascaded design of some loops, the setpoint of the inner-loop can itself become oscillatory as result of the stiction. In particular, Fig. 11(a) to 11(c) show that loops CHEM_02, CHEM_19 and CHEM_22 exhibit a distinct pattern in the ER and OP signals. To emulate this behaviour, a sine wave with amplitude, phase and frequency identical to those defined for the sinusoidal disturbance case in Table 2 is added to the original setpoint. When the frequencies of the setpoint and stiction oscillations are aligned, the observed pattern can indeed be reproduced (see Fig. 11(d)).

### 3.1.5. Stiction model

The model used to create the stiction behaviour is the XCH model of Xie et al. (2013), an improved version of the widely-used model of Choudhury et al. (2005). The modifications to the original model address the issues identified in Garcia (2008), which are seen when the model is subjected to the control valves tests set by the International Society of Automation.

The deadband + stickband parameter $S$ and slip-jump parameter $J$ described in Fig. 2 are each randomly chosen from the range $[1 : 0.1 : 10]$. Therefore the stiction simulations cover the three cases described in Choudhury et al. (2005):

- $S < J$ (overshoot);
- $S > J$ (undershoot);
- $S = J$ (no offset).

Loops experiencing deadband ($J = 0$) are not considered here, as this technically differs from stiction and only induces limit cycling behaviour for integrating processes. For all other cases the $S$ and $J$ parameters are set to zero.

An issue not discussed in Amiruddin et al. (2019), Kamaruddin et al. (2020), Henry et al. (2020) is the handling of noisy control signals and their influence on the stiction model. In Choudhury et al. (2005), it is stated that their model is sensitive to noise and recommend placing an exponentially weighted moving average (EWMA) filter between the controller and the stiction model. As the XCH model used here is based on the Choudhury model, it inherits the variable $v_{new} = x(k) - x(k-1)$ and uses it as a means of detecting a change in valve direction. It is therefore likely that the XCH model is also sensitive to noise and so an EWMA filter is placed after the controller for all simulations as suggested.

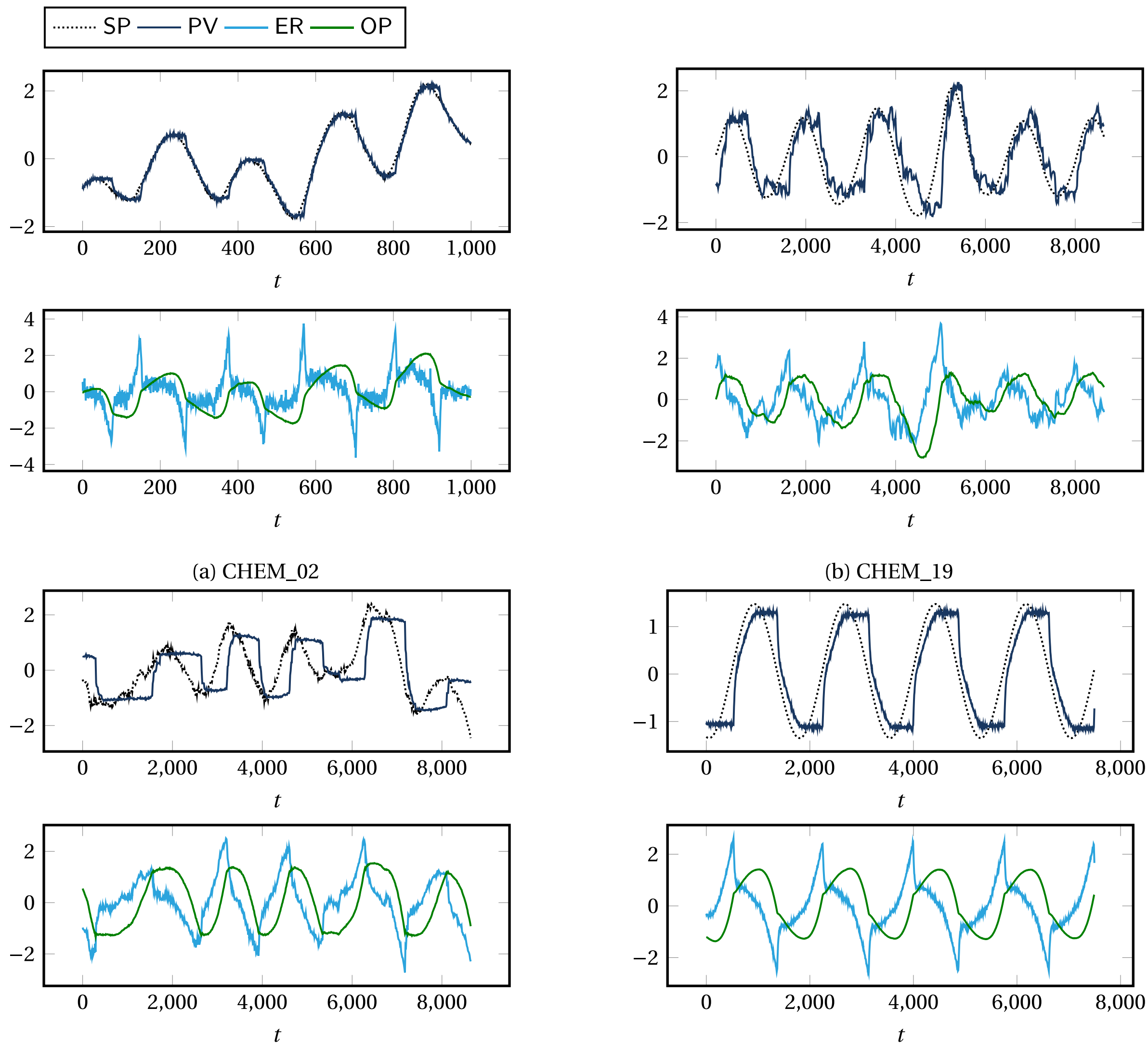
### 3.1.6. Sampling frequency and window size

The rate at which data is recorded is chosen from the following set of values: $[0.1, 1, 3, 5, 10, 12, 15, 20, 30, 60]$ (samples per second). This matches the sampling rates seen in benchmark data of Jelali and Huang (2010). To avoid over/under-sampling, the closed-loop bandwidth $\omega_b$ (rad/s) of the loop (assuming no stiction) is computed, and only cases satisfying the inequality:

$$3\omega_b \leq 1/T_s \leq 10\omega_b, \quad (15)$$

are simulated. This restriction is a modification of the recommended sampling frequencies suggested by Åström and Wittenmark (1997).

Some of the control loops from the industry benchmark include cases with a limited amount of data. It is therefore beneficial to produce a classifier that can be applied to such cases, not only to produce a good benchmark result, but to provide freedom to select shorter windows of data in the event that setpoint changes or disturbances are preventing a




(a) CHEM_02
(b) CHEM_19
(c) CHEM_22
(d) Simulation

**Fig. 11.** Error and controller output signals when stiction is combined with an oscillatory setpoint.

reliable diagnosis. More than 97% of the loops in the benchmark have at least 500 samples available for testing, therefore the simulated data will consist of loops with mixed window sizes, ranging from 500 to 2000 samples.

*3.1.7. Summary*

For the simulated data presented in the following sections there are 3910 cases in total: 2434 loops with stiction (926 of which have an oscillating setpoint), 796 with aggressively tuned controllers, 353 with well-tuned controllers and 327 with sinusoidal disturbance. The ratio of self-regulating to integrating cases is approximately 50-50. A simplified summary of the simulation procedure is provided by the flowchart in Fig. 12.

### 3.2. Feature extraction & pre-processing

Feature extraction is performed in Python using the package `tsfresh` (v0.14.1), developed by Christ et al. (2018). The time series signals used are the error $ER$ and controller output $OP$, both standardised to have zero mean and unit variance. The number of features computed per signal is 794 (the default set of features computed by `tsfresh`), giving 1588 in total. The types of features computed range from common statistical metrics such as mean, median, variance etc., to more abstract features such as the coefficients from a fast Fourier transform or auto-regressive model. It is worth noting however, that these are not totally unique. There are only 63 base features (see Table 9), however many of them are parameterised such that each new parameterised computation constitutes a new feature. For example, the autocorrelation feature is computed for lags 0 through 9, giving 10 distinct inputs to the model. A full and updated list of features, along with details of how the features are extracted can be found in the `tsfresh` documentation (Christ et al., 2020).

When dealing with such a large number of features, it can be easy to overfit to the simulated training data. To mitigate this risk, principal component analysis (PCA) is applied to deal with correlated features and reduce dimensionality. The hypothesis testing element of the `tsfresh` package, used for feature selection (dimensionality reduction), is not utilised in this paper as this did not yield satisfactory results for our dataset. A second and more significant reduction is achieved using linear discriminant analysis (LDA), a supervised technique which maps the input data onto a new set of variables such that the distances between the labelled classes in the new feature-space are maximised. The restriction when using LDA is that the number of components must be less than or equal to min(n_classes - 1, n_features). As there are four

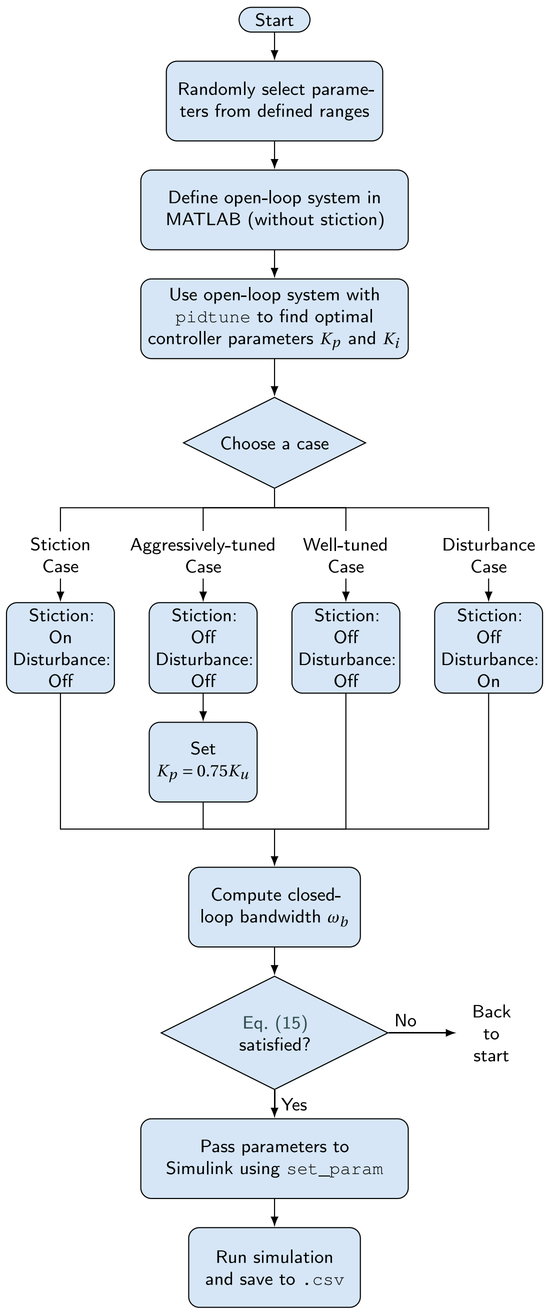



**Fig. 12.** An overview of the data generation process.

classes in this work (stiction, sinusoidal disturbance, aggressive tuning, good tuning), LDA is able to reduce the number of dimensions to three at most. This allows the data and decision function to be visualised in a 3-D plot, which enables improved understanding of the data and the final decision region. The combination of LDA with PCA has shown promise when applied to supervised learning tasks, as demonstrated in Pechenizkiy et al. (2006).

### 3.3. Model training

The features extracted from the simulated data are stored in a $3910 \times 1588$ feature matrix $X$, where each row corresponds to a unique simulation and each column a computed feature. For every row in $X$ there is an associated class label stored in a vector $y$, with labels corresponding one of the four simulation cases. The feature matrix

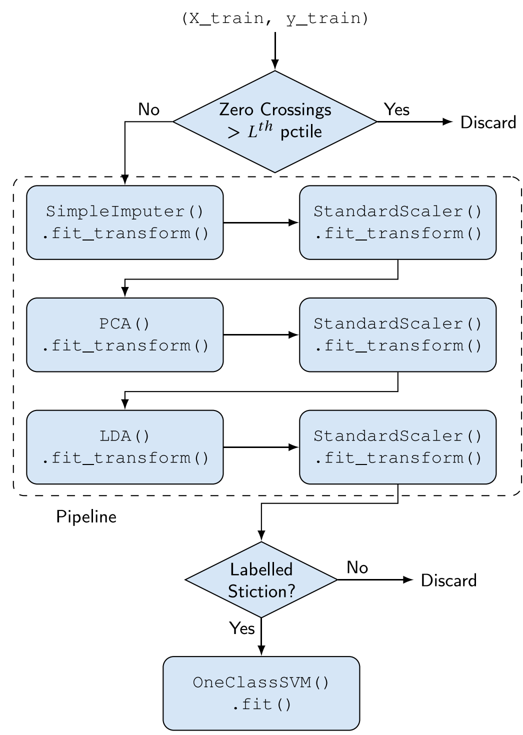



**Fig. 13.** Pre-processing pipeline during training.

and class labels are split into three sets: 70% of the data is used for training `(X_train, y_train)`, 15% for validation `(X_val, y_val)` and the remaining 15% for testing `(X_test, y_test)`. All data is stratified by the label such that each of the four cases are represented equally across the datasets. The classifier is the scikit-learn (v0.22.1) implementation of the one-class support vector machine with RBF kernel, discussed in Section 2.3. The $\gamma$ parameter in the RBF kernel (Eq. (4)) is defined using the default option `1/(n_features)`. The $\nu$ parameter of the one-class SVM (Eq. (7)) determines how much of the training data can be regarded as outlying; in order to achieve a more generalised decision boundary, it is considered as a hyperparameter which is optimised as part of the training process. The full training process is shown in Fig. 13, and the steps are detailed below:

Step 1 Begin with `X_train`, a DataFrame of the features extracted from the simulated data.
Step 2 Remove cases where number of zero-crossings (computed by `tsfresh`) exceeds some percentile $L$.
Step 3 Impute the data to ensure there are no missing values for any of the features (replace with zeros).
Step 4 Scale the extracted features using the mean and standard deviation of
Step 5 Compute the desired number of principal components of the data to decorrelate the data and reduce dimensionality.
Step 6 Scale the principal components using a second standard scaler.
Step 7 Project the scaled principal components onto 3 dimensions using linear discriminant analysis.
Step 8 Scale the LDA features using a third standard scaler.
Step 9 Remove any non-stiction examples, as the One-Class SVM is an unsupervised method and requires only the stiction data. Note that this step is reserved till last as LDA requires all four classes to transform the data.

Steps 3-8 can be combined into a single Pipeline object, as all future predictions need to undergo an identical scaling and transformation process. With the training data, `.fit(X_train, y_train)` is used

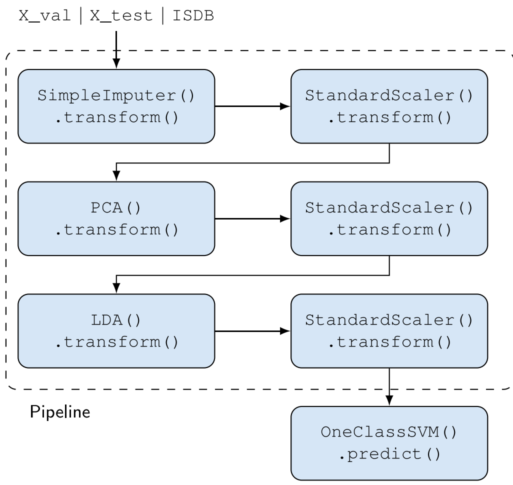



**Fig. 14.** Fitted pipeline used to transform new data.

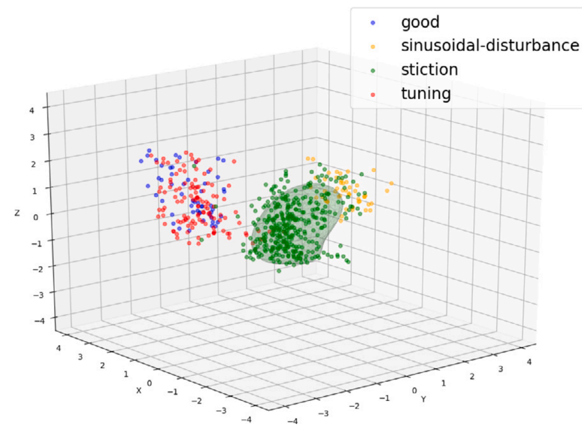



**Fig. 15.** Decision region of the one-class SVM (Model 1), along with simulated testing data coloured by label.

**Table 3**
Parameters considered in the grid search.

| Parameter | Values |
|---|---|
| `nu` | [0.01, 0.02, …, 0.40] |
| $L$ | [0.70, 0.71, …, 0.90] |
| `n_components` | [0.9900, 0.9990, 0.9999] |

to determine all of the necessary parameters to perform the scaling and transformation, and `.transform(X_train)` is used to apply them. For convenience, `.fit_transform(X_train, y_train)` performs the fit and transform actions sequentially. All non-training data use just the transform method i.e. `.transform(X_test)` (see Fig. 14).

The hyperparameters considered for the model are selected from Table 3. The default parameters for each component in the pipeline are listed in Table 10. The `nu` parameter for the one-class SVM determines how many examples in the data can be considered as outliers; a value of 0.10 would assume that 10% of the data are outliers and use the remaining 90% to create the decision function. The `n_components` parameter of the PCA transform determines how many principal components are used in the transform, this can be a number or a fraction. If a fraction is used, then the number of components is chosen such that the amount of variance that needs to be explained is greater than the percentage specified by the fraction. For example, `n_components =0.9999` gives 1109 principal components, which is the number needed to explain 99.99% of the variance of the data. The $L$ parameter is the upper limit of the number of zero-crossings in the error signal, where $L = 0.7$ implies that any training examples where the number of crossings exceed the 70th percentile of `X_train` are removed. This step was added after it was discovered that filtering `X_train` in this way had a significant impact on the overall performance of the model, this is discussed further in Section 4.2. To ensure that such cases are still classified correctly, this filtering is not applied to the testing and validation data.

## 4. Results and comparisons

In this section a collection of optimised stiction classification models are presented. The top performing models are applied to loops from the "international stiction database" (ISDB) benchmark data found in Jelali and Huang (2010). The best result is compared with a number of existing methods, including a handful of newly developed learning-based solutions.

### 4.1. Simulation results

With a trained model, new predictions can be made via the process demonstrated in Fig. 16. As there are three latent features produced by LDA, they can be plotted with the 3-dimensional decision region derived from the SVM. This is seen in Fig. 15, which shows a scatter plot of the simulated test data in the new feature space. All subsequent predictions are labelled as either true positive (TP), true negative (TN), false positive (FP) or false negative (FN). The metrics used to evaluate performance are as defined as follows:

$$\text{Precision} = \frac{(TP)}{(TP + FP)} \tag{16}$$

$$\text{Recall} = \frac{(TP)}{(TP + FN)} \tag{17}$$

$$\text{F1-Score} = \frac{2TP}{2TP + FP + FN} \tag{18}$$

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{19}$$

Note that balanced accuracy is preferred to regular accuracy to account for differences in the number of stiction versus non-stiction cases. To derive the best results, the model is trained using different values for the hyperparameters discussed earlier and the performance is recorded. As most existing methods show a preference for false positive predictions over false negative, there is some distrust when it comes to the reliability of existing stiction detection methods. To address this the validation results are sorted by prioritising precision over recall. The parameters and results of the top 10 selected models are given in Table 4.

As the training set consists of only stiction cases, the precision, F1-score and balanced accuracy metrics are omitted for this set. Model 1 provides the highest precision and F1-score on the validation set and is considered to be the best result. This is also mirrored in the test data, which implies the selected parameters have not led to overfitting.

### 4.2. Industrial benchmark results

Next, the classifier is applied to 81 industrial control loops from the benchmark dataset presented in Jelali and Huang (2010). As evidenced by Fig. 17, the classifier performs best on the simulated data when applied to window lengths between 750–1500. For this reason, window lengths of 1000 are preferred when testing the industry data. For consistency, where possible, any data used in testing is either identical to, or a subset of, the window used to evaluate the bicoherence and ellipse fitting method of Choudhury et al. (2004) (BIC), as it is the highest performing method in the book. BIC is restricted to loops with

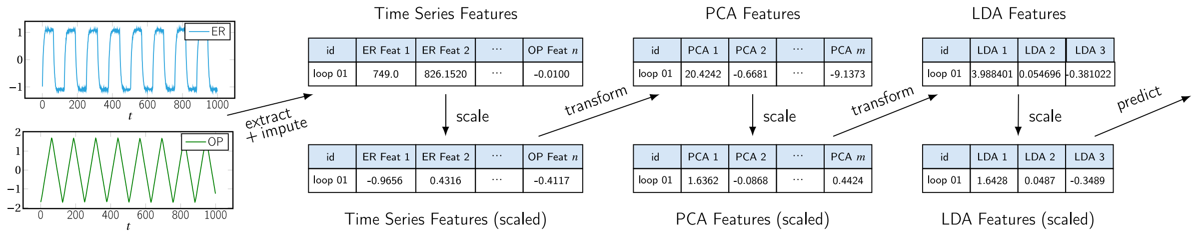



**Fig. 16.** An artificial example of the time series feature extraction, scaling and dimensionality reduction pipeline.

**Table 4**
Results of the top 10 models (with $L \leq 0.9$) when applied to simulated data, sorted by precision and recall on the validation set.

| Model | Parameters | | | Training ($n = 1651$–$1687$) | | | | Validation ($n = 586$) | | | | Testing ($n = 587$) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | n_comp | $L$ | nu | Prec. | Rec. | F1 | Bal. | Prec. | Rec. | F1 | Bal. | Prec. | Rec. | F1 | Bal. |
| 1 | 0.9999 | 0.70 | 0.34 | NA | 0.660 | NA | NA | 0.996 | 0.658 | 0.792 | 0.827 | 0.996 | 0.630 | 0.772 | 0.813 |
| 2 | 0.9999 | 0.75 | 0.37 | NA | 0.630 | NA | NA | 0.996 | 0.644 | 0.782 | 0.820 | 0.996 | 0.625 | 0.768 | 0.810 |
| 3 | 0.9999 | 0.70 | 0.35 | NA | 0.651 | NA | NA | 0.996 | 0.638 | 0.778 | 0.817 | 1.000 | 0.614 | 0.761 | 0.807 |
| 4 | 0.9999 | 0.75 | 0.38 | NA | 0.620 | NA | NA | 0.996 | 0.633 | 0.774 | 0.814 | 0.996 | 0.616 | 0.761 | 0.806 |
| 5 | 0.9999 | 0.70 | 0.36 | NA | 0.639 | NA | NA | 0.996 | 0.630 | 0.772 | 0.813 | 1.000 | 0.608 | 0.756 | 0.804 |
| 6 | 0.9999 | 0.70 | 0.37 | NA | 0.630 | NA | NA | 0.996 | 0.630 | 0.772 | 0.813 | 1.000 | 0.597 | 0.748 | 0.799 |
| 7 | 0.9999 | 0.71 | 0.37 | NA | 0.630 | NA | NA | 0.996 | 0.625 | 0.768 | 0.810 | 0.995 | 0.592 | 0.742 | 0.794 |
| 8 | 0.9990 | 0.75 | 0.39 | NA | 0.611 | NA | NA | 0.996 | 0.619 | 0.764 | 0.807 | 0.995 | 0.603 | 0.751 | 0.799 |
| 9 | 0.9999 | 0.75 | 0.39 | NA | 0.609 | NA | NA | 0.996 | 0.619 | 0.764 | 0.807 | 1.000 | 0.605 | 0.754 | 0.803 |
| 10 | 0.9999 | 0.70 | 0.38 | NA | 0.619 | NA | NA | 0.996 | 0.616 | 0.761 | 0.806 | 1.000 | 0.592 | 0.744 | 0.796 |

**Table 5**
Industry benchmark results, all windows (left) and average result per loop (right).

| Model | Industry Benchmark ($n = 378$) | | | | | | | | | Average Industry Benchmark ($n = 81$) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | TN | FP | FN | Correct | Prec. | Rec. | F1 | Bal. | TP | TN | FP | FN | Correct | Prec. | Rec. | F1 | Bal. |
| 1 | 92 | 213 | 19 | 54 | 305 | 0.829 | 0.630 | 0.716 | 0.774 | 24 | 42 | 3 | 12 | 66 | 0.889 | 0.667 | 0.762 | 0.800 |
| 2 | 78 | 219 | 13 | 68 | 297 | 0.857 | 0.534 | 0.658 | 0.739 | 19 | 42 | 3 | 17 | 61 | 0.864 | 0.528 | 0.655 | 0.731 |
| 3 | 91 | 215 | 17 | 55 | 306 | 0.843 | 0.623 | 0.717 | 0.775 | 24 | 42 | 3 | 12 | 66 | 0.889 | 0.667 | 0.762 | 0.800 |
| 4 | 78 | 219 | 13 | 68 | 297 | 0.857 | 0.534 | 0.658 | 0.739 | 19 | 42 | 3 | 17 | 61 | 0.864 | 0.528 | 0.655 | 0.731 |
| 5 | 89 | 215 | 17 | 57 | 304 | 0.840 | 0.610 | 0.706 | 0.768 | 24 | 42 | 3 | 12 | 66 | 0.889 | 0.667 | 0.762 | 0.800 |
| 6 | 87 | 216 | 16 | 59 | 303 | 0.845 | 0.596 | 0.699 | 0.763 | 23 | 42 | 3 | 13 | 65 | 0.885 | 0.639 | 0.742 | 0.786 |
| 7 | 86 | 215 | 17 | 60 | 301 | 0.835 | 0.589 | 0.691 | 0.758 | 22 | 42 | 3 | 14 | 64 | 0.880 | 0.611 | 0.721 | 0.772 |
| 8 | 81 | 223 | 9 | 65 | 304 | 0.900 | 0.555 | 0.686 | 0.758 | 19 | 42 | 3 | 17 | 61 | 0.864 | 0.528 | 0.655 | 0.731 |
| 9 | 76 | 220 | 12 | 70 | 296 | 0.864 | 0.521 | 0.650 | 0.734 | 18 | 42 | 3 | 18 | 60 | 0.857 | 0.500 | 0.632 | 0.717 |
| 10 | 87 | 217 | 15 | 59 | 304 | 0.853 | 0.596 | 0.702 | 0.766 | 23 | 42 | 3 | 13 | 65 | 0.885 | 0.639 | 0.742 | 0.786 |

at least 1000 data points, and as a result there are several cases where the BIC method has not been applied due to an insufficient number of samples; in these instances, the full length of data is used in this work (see Table 11 for details).

To address intermittent oscillations and to demonstrate consistency, where possible, multiple windows from the same loop are tested. Table 5 contains the results when applied to 378 different time series windows from the benchmark data. In the event that windows from the same loop produce conflicting results, the most common verdict is taken (stiction is assumed if 50-50). This would mimic the recommended use of this classifier when applied in a real system, as only loops which demonstrate consistent stiction behaviour should be flagged for investigation.

The model which produced the best result on the simulated data performs comparably here. Small differences are expected due to disturbances, insufficient data and infrequent sampling, but generally performance is quite similar. Optimising for precision on the validation data has a corresponding effect on the industrial data, as false positives appear minimal. Note that the non-averaged results of Model 1 in Table 5 show 19 false positives, which may seem excessive but upon further inspection 13 of these can be attributed to just two loops (PAP_09, CHEM_27). However infrequent, false positives can occasionally appear as a result of process disturbances, which is one reason why the average prediction is the preferred metric (see Fig. 18).

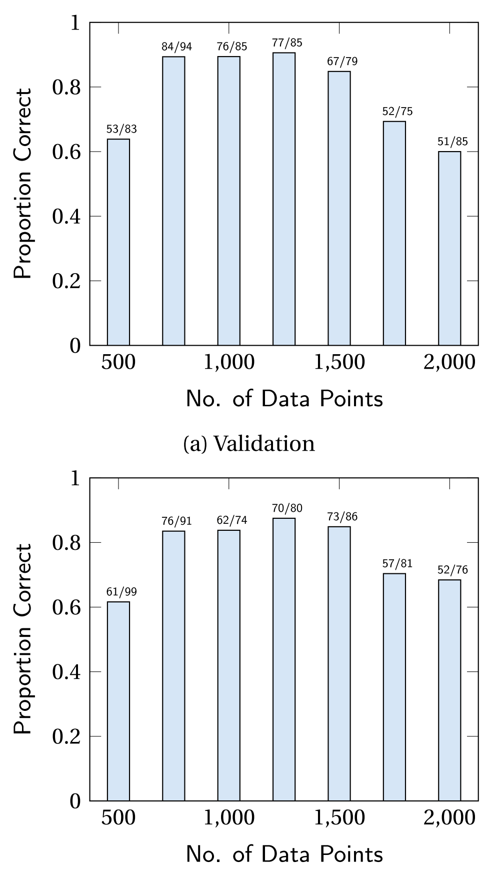
In Table 3 note that $L \leq 0.9$, if this restriction is not imposed when searching for optimal hyperparameters, then the optimal model found is one with $L \sim 1$, i.e. no filtering on zero-crossings. Without filtering, the performance of each classifier when applied to the simulated test set does not correlate well with the industry dataset, which could mean that the classifiers are overfitting to the simulated data. If the maximum value for $L$ is set to 0.9, the hyperparameter search finds a local optimum with $L = 0.7$–$0.75$, as shown in Table 4. These models perform comparably across both the simulated and real data, and are therefore most suitable for presentation in this section. Both the filtering on zero-crossings discussed here and the data re-sampling discussed below tend to increase the similarity between the training data used and the industrial data, and it is possible that this makes the classification problem easier, leading to enhanced performance on the real-world data. In principle, fixing a maximum value for $L$ to achieve similarity with the industrial data could introduce an element of bias to the classifier, as ideally one would not make changes purely to improve the final classification result. However, given the consistent performance across each dataset and the modest stiction detection rate we believe this bias to be minimal.

### 4.3. Comparison with other detection methods

The performance of Model 1 is compared with the methods described in Jelali and Huang (2010), along with several subsequently developed methods. In Table 6 we see the results of these methods using the metrics in Eqs. (16)–(19). Uncertain verdicts produced by other methods are not included. The one-class SVM (OCSVM) produces the best overall performance with regard to number of correct responses and precision. A balanced accuracy score of 80% falls just behind the

**Table 6**
ISDB benchmark results of 81 loops with comparison against other detection methods.

| | TP | TN | FP | FN | Precision | Recall | F1 Score | Acc. | Bal Acc. | Correct | Verdicts | Applications |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BIC | 23 | 25 | 8 | 1 | 0.742 | 0.958 | 0.836 | 0.842 | 0.858 | 48 | 57 | 57 |
| CORR | 13 | 7 | 5 | 1 | 0.722 | 0.929 | 0.813 | 0.769 | 0.756 | 20 | 26 | 49 |
| HIST | 21 | 10 | 18 | 8 | 0.538 | 0.724 | 0.618 | 0.544 | 0.541 | 31 | 57 | 65 |
| RELAY | 27 | 7 | 27 | 3 | 0.500 | 0.900 | 0.643 | 0.531 | 0.553 | 34 | 64 | 69 |
| CURVE | 22 | 5 | 16 | 7 | 0.579 | 0.759 | 0.657 | 0.540 | 0.498 | 27 | 50 | 63 |
| AREA | 15 | 15 | 5 | 13 | 0.750 | 0.536 | 0.625 | 0.625 | 0.643 | 30 | 48 | 48 |
| HAMM2 | 31 | 9 | 32 | 4 | 0.492 | 0.886 | 0.633 | 0.526 | 0.553 | 40 | 76 | 76 |
| HAMM3 | 33 | 11 | 34 | 3 | 0.493 | 0.917 | 0.641 | 0.543 | 0.581 | 44 | 81 | 81 |
| SLOPE | 22 | 7 | 13 | 9 | 0.629 | 0.710 | 0.667 | 0.569 | 0.530 | 29 | 51 | 61 |
| ZONES | 19 | 9 | 14 | 11 | 0.576 | 0.633 | 0.603 | 0.528 | 0.512 | 28 | 53 | 61 |
| NLPCA-AC | 20 | 34 | 6 | 15 | 0.769 | 0.571 | 0.656 | 0.720 | 0.711 | 54 | 75 | 78 |
| SIGMOID | 11 | 6 | 1 | 2 | 0.917 | 0.846 | 0.880 | 0.850 | 0.852 | 17 | 20 | 20 |
| KMEANS | 13 | 3 | 4 | 0 | 0.765 | 1.000 | 0.867 | 0.800 | 0.714 | 16 | 20 | 20 |
| SDN | 26 | 35 | 9 | 10 | 0.743 | 0.722 | 0.732 | 0.763 | 0.759 | 61 | 80 | 80 |
| BSD | 12 | 8 | 3 | 3 | 0.800 | 0.800 | 0.800 | 0.769 | 0.764 | 20 | 26 | 26 |
| CNN-PCA | 27 | 28 | 14 | 9 | 0.659 | 0.750 | 0.701 | 0.705 | 0.708 | 55 | 78 | 78 |
| MTSFCC | 13 | 9 | 2 | 2 | 0.867 | 0.867 | 0.867 | 0.846 | 0.842 | 22 | 26 | 26 |
| OCSVM | 24 | 42 | 3 | 12 | 0.889 | 0.667 | 0.762 | 0.815 | 0.800 | 66 | 81 | 81 |
| OCSVM* | 27 | 42 | 3 | 9 | 0.900 | 0.750 | 0.818 | 0.852 | 0.842 | 69 | 81 | 81 |



(a) Validation

(b) Testing

**Fig. 17.** Number of correct responses per window length on simulated validation and testing data.

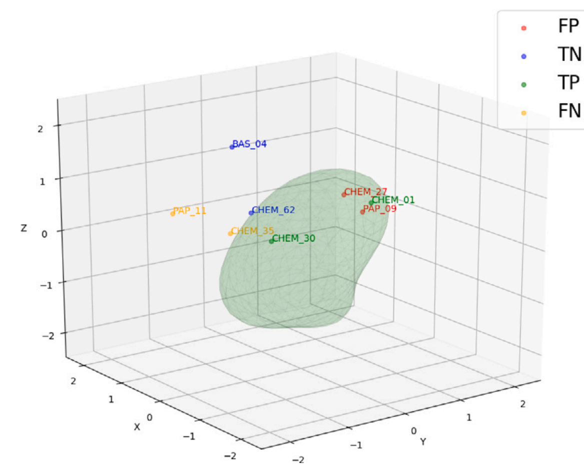



**Fig. 18.** A selection of loops from the ISDB benchmark data.

BIC method, which is able to achieve 86%, although its application is limited to 57 loops in the database. When applied to only those loops, the new method ties for 50/57 with the stiction detection network (SDN) method of Amiruddin et al. (2019).

As many of the benchmark loops have not been officially diagnosed, it is possible that some of the suspected diagnoses are incorrect. In Table 7, the results when applied to only the 26 loops with confirmed faults are presented. The new method also produces the most correct responses here.

### 4.4. Analysis and discussion

It is important that the misdiagnosed loops be studied to try to identify any shared behaviour, as they may provide some insight into how to improve results for future iterations. In total, 27 cases of stiction were diagnosed by Model 1, just 3 of which were false positive:

- CHEM_27 demonstrates a clear oscillatory pattern, although not the sharp triangular signal expected of a level loop. The classifier has no knowledge of the process type, so an incorrect result may stem from the confusion between self-regulating and integrating training data.
- PAP_04 is listed as suffering from "deadzone + tight tuning". Whilst deadzone can induce oscillations in integrating loops, as this is a self-regulating process the oscillations are possibly a symptom of the poor tuning. Manual inspection of the loop shows that the oscillations are dissimilar to the aggressively tuned training data, and instead more closely resemble a stiction case with low resolution sampling.
- PAP_09 is simply labelled as "no stiction". This loop is amongst those with known faults, but with no detailed explanation behind

**Table 7**
ISDB benchmark results of the 26 loops with known faults.

| | TP | TN | FP | FN | Precision | Recall | F1 Score | Acc. | Bal Acc. | Correct | Verdicts | Applications |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BIC | 14 | 4 | 5 | 0 | 0.737 | 1.000 | 0.848 | 0.783 | 0.722 | 18 | 23 | 23 |
| CORR | 9 | 5 | 3 | 0 | 0.750 | 1.000 | 0.857 | 0.824 | 0.813 | 14 | 17 | 23 |
| HIST | 11 | 5 | 5 | 3 | 0.688 | 0.786 | 0.733 | 0.667 | 0.643 | 16 | 24 | 25 |
| RELAY | 14 | 3 | 7 | 0 | 0.667 | 1.000 | 0.800 | 0.708 | 0.650 | 17 | 24 | 26 |
| CURVE | 11 | 3 | 4 | 3 | 0.733 | 0.786 | 0.759 | 0.667 | 0.607 | 14 | 21 | 25 |
| AREA | 8 | 3 | 2 | 5 | 0.800 | 0.615 | 0.696 | 0.611 | 0.608 | 11 | 18 | 18 |
| HAMM2 | 14 | 4 | 7 | 1 | 0.667 | 0.933 | 0.778 | 0.692 | 0.648 | 18 | 26 | 26 |
| HAMM3 | 14 | 3 | 8 | 1 | 0.636 | 0.933 | 0.757 | 0.654 | 0.603 | 17 | 26 | 26 |
| SLOPE | 12 | 3 | 2 | 3 | 0.857 | 0.800 | 0.828 | 0.750 | 0.700 | 15 | 20 | 25 |
| ZONES | 10 | 6 | 2 | 4 | 0.833 | 0.714 | 0.769 | 0.727 | 0.732 | 16 | 22 | 25 |
| NLPCA-AC | 8 | 8 | 3 | 7 | 0.727 | 0.533 | 0.615 | 0.615 | 0.630 | 16 | 26 | 26 |
| SIGMOID | 11 | 6 | 1 | 2 | 0.917 | 0.846 | 0.880 | 0.850 | 0.852 | 17 | 20 | 20 |
| KMEANS | 13 | 3 | 4 | 0 | 0.765 | 1.000 | 0.867 | 0.800 | 0.714 | 16 | 20 | 20 |
| SDN | 11 | 8 | 3 | 4 | 0.786 | 0.733 | 0.759 | 0.731 | 0.730 | 19 | 26 | 26 |
| BSD | 12 | 8 | 3 | 3 | 0.800 | 0.800 | 0.800 | 0.769 | 0.764 | 20 | 26 | 26 |
| CNN-PCA | 12 | 9 | 2 | 3 | 0.857 | 0.800 | 0.828 | 0.808 | 0.809 | 21 | 26 | 26 |
| MTSFCC | 13 | 9 | 2 | 2 | 0.867 | 0.867 | 0.867 | 0.846 | 0.842 | 22 | 26 | 26 |
| OCSVM | 12 | 9 | 2 | 3 | 0.857 | 0.800 | 0.828 | 0.808 | 0.809 | 21 | 26 | 26 |
| OCSVM* | 13 | 9 | 2 | 2 | 0.867 | 0.867 | 0.867 | 0.846 | 0.842 | 22 | 26 | 26 |

this verdict it is difficult to provide a reason for the incorrect result.

Of the 54 negative predictions there are 12 loops which were falsely identified as not having stiction:

- BAS_06 has less than the preferred 1000 samples, but the issue appears to be the frequency of oscillation. A correct result can be achieved by manually upsampling the data and reducing the number of oscillations in a given window.
- CHEM_05 has far too few samples. Upsampling can be used to obtain a correct result, but it is risky due to the amount of data that needs to be inferred.
- CHEM_07 and CHEM_08 are in manual mode, so a regular stiction pattern cannot be identified. Like BIC, loops in manual control would be excluded from testing in practice, however the results are presented to provide a worst case scenario.
- CHEM_12 has very few oscillations in a 1000 sample window, this can be addressed by using the full 2000 samples and downsampling.
- CHEM_19 exhibits the pattern shown in Fig. 11 and should be detected, however there is a disturbance in the data which may be interfering with the diagnosis.
- CHEM_20 shows very fast oscillations due to stiction. However, the controller appears slightly sluggish, so the dominant oscillation in the error signal is due to the oscillating setpoint, making the stiction oscillations seem more noise-like.
- CHEM_25 shows a clear stiction pattern but with frequent oscillations that may be tuning related. This theory agrees with the original comments accompanying the loop, which suggest that the loop is "possibly marginally stable". A correct verdict can be achieved through manually upsampling.
- CHEM_29 also exhibits similar patterns to Fig. 11, however there are too many oscillations in a 1000 sample window. Upsampling halves the number of oscillations per window, and leads to a correct result.
- PAP_01 shows the classic stiction pattern, but contains barely two full oscillation cycles. With more data this is expected to produce a correct result.
- PAP_11 indeed shows an underlying stiction pattern, but its appearance is masked by the large spikes in the error signal. These spikes are a common occurrence in the aggressively tuned loops, so it is not surprising that this loop lies amongst the tuning cases when added to a plot such as Fig. 15.

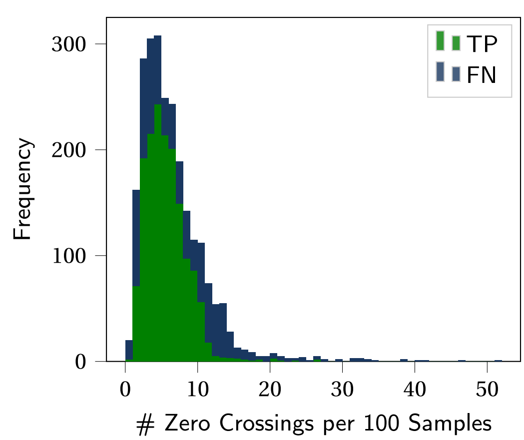



**Fig. 19.** Frequency distribution (stacked) of predictions for simulated stiction cases.

- POW_01 has fast oscillations and sampling is fairly low resolution. Manually upsampling helps a little, but not enough to produce an overall correct result.

The findings above indicate that there is indeed a pattern with the missed stiction cases, as many of them can be correctly detected through manually upsampling or downsampling. This suggests the classifier is sensitive to the number of oscillations in a given window.

*4.4.1. Automated data re-sampling*

The idea of including re-sampling as part of the data pre-processing is explored further by studying the true positive/false negative prediction rates on the simulated stiction data. It can be seen in Fig. 19 that loops with fewer than 1 or greater than 12 zero-crossings per 100 samples are very likely to be classified incorrectly. This provides further evidence for the theory that the classifier is sensitive to oscillation frequency. Loops BAS_06, CHEM_05, CHEM_25, CHEM_29 and POW_01 all benefit from upsampling, as this can make up for insufficient data and reduce the number of oscillations in a given window. Conversely, loops such as CHEM_12 and PAP_01 (if more data was available) could benefit from downsampling i.e. increasing the number of oscillations per window. There are also a handful of non-stiction cases where

**Table 8**
Details of 10 control loops supplied by Phillips 66 Ltd and our OCSVM and OCSVM* classification results (verdict given in parentheses). The range of data used, and the resampling rates used for the OCSVM* results, are also shown. The resampling rate, obtained using the algorithm shown in Fig. 20, is either 0.5 (upsampled with linear interpolation), 1 (original data) or 2 (sampled every other data point).

| NAME | INDUSTRY | LOOP TYPE | SAMPLES | SAMPLE_FREQ | COMMENTS | WINDOW | RESAMPLE | STICTION | OCSVM | OCSVM* |
|---|---|---|---|---|---|---|---|---|---|---|
| P66_01 | Chemical | Level | 8588 | [8–51] | Stiction | 1–8588 | 1 | YES | 11/16 (YES) | 11/16 (YES) |
| P66_02 | Chemical | Temperature | 16383 | [4–20] | Stiction | 1–16383 | 1 | YES | 29/31 (YES) | 29/31 (YES) |
| P66_03 | Chemical | Pressure | 8635 | [18–41] | Stiction | 1–8635 | 0.5 | YES | 0/16 (NO) | 30/33 (YES) |
| P66_04 | Chemical | Flow | 16830 | [4–20] | Stiction | 1–16830 | 1 | YES | 19/32 (YES) | 19/32 (YES) |
| P66_05 | Chemical | Pressure | 5248 | [19–180] | Stiction | 1–5248 | 1 | YES | 0/9 (NO) | 0/9 (NO) |
| P66_06 | Chemical | Flow (Turbine Meter) | 12475 | [5–30] | Stiction | 1–12475 | 1 | YES | 1/23 (NO) | 1/23 (NO) |
| P66_07 | Chemical | Flow | 17035 | [4–15] | Stiction | 1–17035 | 1 | YES | 26/33 (YES) | 26/33 (YES) |
| P66_08 | Chemical | Flow (Turbine Meter) | 12969 | [4–45] | Stiction | 1–12969 | 1 | YES | 0/24 (NO) | 0/24 (NO) |
| P66_09 | Chemical | Flow | 17152 | [5–15] | Stiction | 1–17152 | 1 | YES | 2/33 (NO) | 2/33 (NO) |
| P66_10 | Chemical | Level | 16092 | [4–20] | Stiction | 1–16092 | 2 | YES | 6/31 (NO) | 10/15 (YES) |

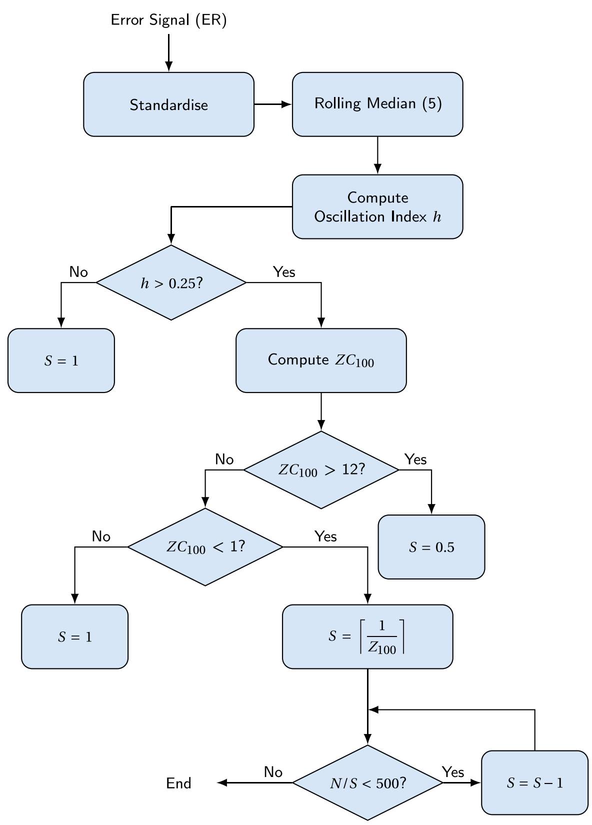



**Fig. 20.** Data resample selection flow chart.

not enough oscillations are captured, such as BAS_01, which could be receiving a free "no stiction" verdict as a result. Resampling such loops would provide a fairer assessment of the predictive capabilities of the model.

The problem of determining a suitable window and sampling rate has been raised previously in Amiruddin et al. (2019), but there is as of yet no attempt at automating this process. Here, a simple method for determining a suitable resampling rate is presented. The method uses the oscillation index of Forsman and Stattin (1999) to identify whether the loop is sufficiently oscillatory. Since the index is heavily dependent on the number of zero-crossings, the 5-point rolling median is computed first to reduce the influence of noise. A value less than 0.25 for the index means that the loop shows very little evidence of oscillation, so resampling is not necessary. If the index is greater than 0.25 then the average number of zero-crossings per 100 data points ($Z_{100}$) is computed, and a sampling rate is chosen such that this value falls within a desired range. Loops exceeding this range use a single pass of linear interpolation to upsample the data. Whereas loops falling short of this range are downsampled until either the number of zero-crossings per 100 samples is greater than one, or until there is not enough data to downsample further (500 minimum). The flow diagram for this process is seen in Fig. 20.

The results when combining this resampling process with Model 1 are recorded as OCSVM* in Tables 6, 7 and 11. The verdicts for loops BAS_06, CHEM_12 and CHEM_25 switch from "no stiction" to "stiction", increasing the number of correct detections by 3 without introducing any new false positives. In addition, loops such as BAS_01, CHEM_24 and CHEM_44 retain their correct responses despite the change in sampling. The verdict for CHEM_05, originally false negative, stays consistent despite upsampling; the reason for this is possibly insufficient data as there are still only 400 data points after upsampling.

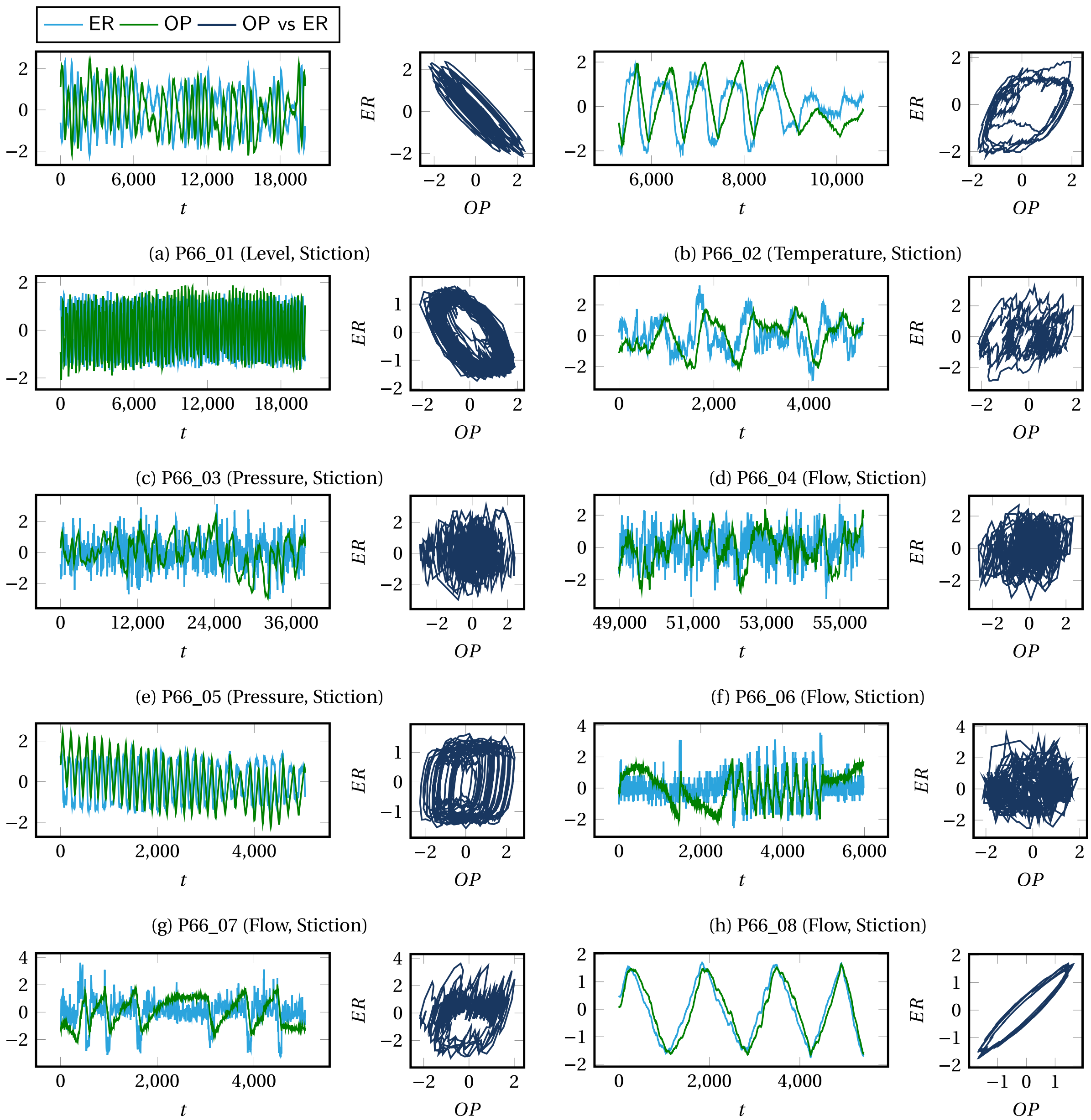
### 4.5. Additional industry data

Normalised OP and ER data taken from 10 new control loops from the chemical industry are seen in Fig. 21. These loops are subjected to classification both with and without the automated re-sampling, and the results are seen in Table 8. Each control loop is labelled as stiction, however it is not known whether this has been officially confirmed with a full valve test. Without re-sampling, the OCSVM method identifies just four cases of stiction: P66_01, P66_02, P66_04 and P66_07. Manual inspection of loops P66_05, P66_06, P66_08 and P66_09 show that excessive noise in the error signal is most likely the primary contributor to these false diagnoses. It is worth noting that these loops fail to show the butterfly-like pattern discussed in Kamaruddin et al. (2020), so it is possible that this method would also misdiagnose these cases. The remaining loops, P66_03 and P66_10, display a clear stiction pattern which is undetected due to the number of oscillations present per window. Running each loop through the re-sampling process described in Fig. 20 yields re-sampling consistent with that which would have been chosen manually. That is, upsampling is suggested for P66_03 and downsampling for P66_10. The remaining 8 loops are left unchanged, bringing the total number of correct responses to 6.

## 5. Conclusions

In this study, a one-class SVM is trained using a variety of simulated process models. Using the PCA-LDA transformed time series features of `tsfresh`, the method is shown to reliably detect stiction with a low false positive rate. The method demonstrates good accuracy on both simulated and real-world control loops, whilst requiring just the standard OP, PV and SP data. The detection algorithm is easily applied to new examples, as the pre-processing pipeline and classifier can be exported as a single "pickled" python file (Van Rossum, 2020). A simple automated window selection/data re-sampling procedure is also presented; this also improves the reliability and can even be used in conjunction with other methods.

The method is designed for online application, as the features can be extracted from the raw control loop data in real-time. Control loops




(a) P66_01 (Level, Stiction)
(b) P66_02 (Temperature, Stiction)
(c) P66_03 (Pressure, Stiction)
(d) P66_04 (Flow, Stiction)
(e) P66_05 (Pressure, Stiction)
(f) P66_06 (Flow, Stiction)
(g) P66_07 (Flow, Stiction)
(h) P66_08 (Flow, Stiction)
(i) P66_09 (Flow, Stiction)
(j) P66_10 (Level, Stiction)

**Fig. 21.** Normalised time and phase plots for 10 new control loops from the chemical industry (1000 data points each).

**Table 9**
A short description of the features extracted from the time series using `tsfresh`, based on the feature calculators documentation and Christ et al. (2020). A full list of the exact features and parameters is provided as a separate CSV file.

| Feature | Description |
|---|---|
| `abs_energy(x)` | Returns the absolute energy of the time series which is the sum over the squared values in $x$. |
| `absolute_sum_of_changes(x)` | Returns the sum over the absolute value of consecutive changes in the series $x$. |
| `agg_autocorrelation(x, param)` | Calculates the value of an aggregation function (e.g. the variance or the mean) over the autocorrelation for different lags. |
| `agg_linear_trend(x, param)` | Calculates a linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one. |
| `approximate_entropy(x, m, r)` | Computes the vectorised approximate entropy algorithm. |

(continued on next page)

**Table 9** (*continued*).

| Feature | Description |
|---|---|
| `ar_coefficient(x, param)` | This feature calculator fits the unconditional maximum likelihood of an autoregressive AR($k$) process. |
| `augmented_dickey_fuller(x, param)` | The Augmented Dickey–Fuller test is a hypothesis test which checks whether a unit root is present in a time series sample. |
| `autocorrelation(x, lag)` | Calculates the autocorrelation of the specified lag. |
| `binned_entropy(x, max_bins)` | Bins the values of $x$ into max_bins equidistant bins, then calculates the value of $\sum_{k=0}^{\min(\text{max\_bins},\ \text{len}(x))} p_k log(p_k) \cdot \mathbf{1}_{(p_k>0)}$ where $p_k$ is the percentage of samples in bin $k$. |
| `c3(x, lag)` | Uses c3 statistics to measure non linearity in the time series. |
| `change_quantiles(x, ql, qh, isabs, f_agg)` | First fixes a corridor given by the quantiles $q_l$ and $q_h$ of the distribution of $x$. Then calculates the average, absolute value of consecutive changes of the series $x$ inside this corridor. |
| `cid_ce(x, normalize)` | This function calculator is an estimate for a time series complexity (a more complex time series has more peaks and valleys) |
| `count_above_mean(x)` | The number of values in $x$ that are higher than the mean of $x$. |
| `count_below_mean(x)` | The number of values in $x$ that are lower than the mean of $x$. |
| `cwt_coefficients(x, param)` | Calculates a continuous wavelet transform for the Ricker wavelet, also known as the "mexican hat wavelet". |
| `energy_ratio_by_chunks(x, param)` | Calculates the sum of squares of chunk $i$ out of $N$ chunks expressed as a ratio with the sum of squares over the whole series. |
| `fft_aggregated(x, param)` | Returns the spectral centroid (mean), variance, skew, and kurtosis of the absolute fourier transform spectrum. |
| `fft_coefficient(x, param)` | Calculates the fourier coefficients of the one-dimensional discrete Fourier Transform for real input by using the fast fourier transformation algorithm. |
| `first_location_of_maximum(x)` | Returns the first location of the maximum value of $x$. |
| `first_location_of_minimum(x)` | Returns the first location of the minimal value of $x$. |
| `friedrich_coefficients(x, param)` | Coefficients of polynomial $h(x)$, which has been fitted to the deterministic dynamics of the Langevin model. |
| `has_duplicate(x)` | Checks if any value in $x$ occurs more than once. |
| `has_duplicate_max(x)` | Checks if the maximum value of $x$ is observed more than once. |
| `has_duplicate_min(x)` | Checks if the minimal value of $x$ is observed more than once. |
| `index_mass_quantile(x, param)` | The relative index $i$ where $q$% of the mass of the time series $x$ lie left of $i$. |
| `kurtosis(x)` | Returns the kurtosis of $x$ (calculated with the adjusted Fisher–Pearson standardised moment coefficient G2). |
| `large_standard_deviation(x, r)` | Boolean variable denoting if the standard dev of $x$ is higher than $r$ times the range (the difference between max and min of $x$). |
| `last_location_of_maximum(x)` | Returns the relative last location of the maximum value of $x$. |
| `last_location_of_minimum(x)` | Returns the last location of the minimal value of $x$. |
| `length(x)` | Returns the length of $x$. |
| `linear_trend(x, param)` | Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to $\text{len}(x) - 1$. |
| `longest_strike_above_mean(x)` | Returns the length of the longest consecutive subsequence in $x$ that is bigger than the mean of $x$. |
| `longest_strike_below_mean(x)` | Returns the length of the longest consecutive subsequence in $x$ that is smaller than the mean of $x$. |
| `max_langevin_fixed_point(x, r, m)` | Largest fixed point of dynamics $\text{argmax}_x h(x) = 0$ estimated from polynomial $h(x)$, which has been fitted to the deterministic dynamics of the Langevin model. |
| `maximum(x)` | Calculates the highest value of the time series $x$. |
| `mean(x)` | Returns the mean of $x$. |
| `mean_abs_change(x)` | Returns the mean over the absolute differences between subsequent time series values. |
| `mean_change(x)` | Returns the mean over the differences between subsequent time series values. |
| `mean_second_derivative_central(x)` | Returns the mean value of a central approximation of the second derivative. |
| `median(x)` | Returns the median of $x$. |
| `minimum(x)` | Calculates the lowest value of the time series $x$. |
| `number_crossing_m(x, m)` | Calculates the number of crossings of $x$ on $m$. |

(*continued on next page*)

**Table 9** (*continued*).

| Feature | Description |
|---|---|
| `number_cwt_peaks(x, n)` | Number of different peaks in $x$. To estimate the number of peaks, $x$ is smoothed by a ricker wavelet for widths ranging from 1 to $n$. This feature calculator returns the number of peaks that occur at enough width scales and with sufficiently high signal-to-noise-Ratio (SNR). |
| `number_peaks(x, n)` | Calculates the number of peaks of at least support n in the time series $x$. A peak of support $n$ is defined as a subsequence of $x$ where a value occurs, which is bigger than its $n$ neighbours to the left and to the right. |
| `partial_autocorrelation(x, param)` | Calculates the value of the partial autocorrelation function at the given lag. |
| `percentage_of_reoccurring_datapoints_ to_all_datapoints(x)` | Returns the percentage of non-unique data points. |
| `percentage_of_reoccurring_values_to_all_values(x)` | Returns the percentage of values that are present in the time series more than once. |
| `quantile(x, q)` | Calculates the $q$ quantile of $x$. |
| `range_count(x, min, max)` | Count observed values between min and max. |
| `ratio_beyond_r_sigma(x, r)` | Ratio of values that are more than $r\sigma$ away from the mean of $x$. |
| `ratio_value_number_to_time_series_length(x)` | Returns a factor which is 1 if all values in the time series occur only once, and below one if this is not the case. |
| `sample_entropy(x)` | Calculate and return sample entropy of x. |
| `skewness(x)` | Returns the sample skewness of $x$ (calculated with the adjusted Fisher–Pearson standardised moment coefficient G1). |
| `spkt_welch_density(x, param)` | This feature calculator estimates the cross power spectral density of the time series $x$ at different frequencies. |
| `standard_deviation(x)` | Returns the standard deviation of $x$. |
| `sum_of_reoccurring_data_points(x)` | Returns the sum of all data points, that are present in the time series more than once. |
| `sum_of_reoccurring_values(x)` | Returns the sum of all values, that are present in the time series more than once. |
| `sum_values(x)` | Calculates the sum over the time series values. |
| `symmetry_looking(x, param)` | Boolean variable denoting if the distribution of $x$ looks symmetric. |
| `time_reversal_asymmetry_statistic(x, lag)` | Returns the time reversal asymmetry statistic. |
| `value_count(x, value)` | Count occurrences of value in time series $x$. |
| `variance(x)` | Returns the variance of $x$. |
| `variance_larger_than_standard_deviation(x)` | Boolean variable denoting if the variance of $x$ is greater than its standard deviation. |

**Table 10**
Defaults for PCA, LDA, StandardScaler and OneClassSVM.

| Method | Default Options |
|---|---|
| `SimpleImputer()` | `StandardScaler()`<br>`with_mean=True`<br>`with_std=True` |
| `PCA()` | `whiten=False`<br>`svd_solver=''auto''`<br>`tol=0.0`<br>`iterated_power="auto`<br>`random_state=None` |
| `LDA()` | `solver=''svd''`<br>`shrinkage=None`<br>`priors=None`<br>`store_covariance=False`<br>`tol=1e-4` |
| `OneClassSVM()` | `kernel=''rbf''`<br>`degree=3`<br>`gamma=''scale''`<br>`coef0=0.0`<br>`tol=1e-3`<br>`nu=0.5`<br>`shrinking=True`<br>`cache_size=200`<br>`max_iter=-1` |

which demonstrate consistent stiction behaviour can be queued for maintenance during the next scheduled plant shutdown. Quantification post-detection using the calculated `tsfresh` features is yet to be explored and will be a focus of future research. The classification tool is written entirely in Python and can be implemented on low-cost edge devices such as a Raspberry Pi. Further optimisations on the window length and number of features may be necessary to improve speed, as this is the primary drawback of the method. Of course, caution is always advised when applying automated systems; manual inspection and further diagnostic tests are recommended before fully committing to valve replacement/repairs.

With regards to future research there are a number of directions to take that could advance the field. Machine learning based stiction diagnosis/quantification is still in its infancy but appears to produce very promising results. As demonstrated by Henry et al. (2020), techniques that have proven useful in other domains such as the bayesian network approaches for fault detection in Kumari et al. (2022b,a), can be adapted to fit the stiction problem. Additional input features such as control-specific performance indices may also provide a boost to performance. Window selection and data pre-processing can also yield improved results, as demonstrated by the simple procedure implemented in this work. There is also the issue of data, as until such time that there is a vast amount of real data to both train and test machine learning approaches, there will always be improvements that could be made to the simulation. A universal simulated training/testing set would be useful to compare different learning-based approaches, as

**Table 11**
Detailed ISDB benchmark information from Jelali and Huang (2010) and our OCSVM and OCSVM* results. The range of data used, and the resampling rates used for the OCSVM* results, are also shown. The resampling rate, obtained using the algorithm shown in Fig. 20, is either 0.5 (upsampled with linear interpolation), 1 (original data) or 2 (sampled every other data point). The selection of data windows used for our analysis is explained in Section 4.2.

| NAME | LOOP TYPE | SAMPLES | SAMPLE_FREQ | COMMENTS | WINDOW | RESAMPLE | STICTION | OCSVM | OCSVM* |
|---|---|---|---|---|---|---|---|---|---|
| BAS_01 | Temperature | 277115 | 1 | No oscillation | 77001–80000 | 2 | NO | NO | NO |
| BAS_02 | Temperature | 277115 | 1 | No oscillation | 35001–39096 | 1 | NO | NO | NO |
| BAS_03 | Temperature | 246827 | 3 | Intermittent oscillation | 30001–34096 | 1 | NO | NO | NO |
| BAS_04 | Pressure | 246827 | 3 | Intermittent oscillation | 77001–81096 | 1 | NO | NO | NO |
| BAS_06 | Temperature | 501 | 1 | Stiction and tight tuning | 1–501 | 0.5 | YES | NO | YES |
| BAS_07 | Temperature | 560 | 1 | Stiction | 1–560 | 1 | YES | YES | YES |
| BAS_08 | Temperature | 42512 | 60 | No oscillation | 1–20000 | 1 | NO | NO | NO |
| CHEM_01 | Flow | 1625 | 1 | Stiction | 1–1625 | 1 | YES | YES | YES |
| CHEM_02 | Flow | 1000 | 1 | Stiction | 1–1000 | 1 | YES | YES | YES |
| CHEM_03 | Temperature | 1945 | 30 | Quantisation | 1–1945 | 1 | NO | NO | NO |
| CHEM_04 | Level | 200 | 1 | Tuning problem | 1–200 | 1 | NO | NO | NO |
| CHEM_05 | Flow | 201 | 1 | Stiction | 1–201 | 0.5 | YES | NO | NO |
| CHEM_06 | Flow | 1000 | 1 | Stiction | 1–1000 | 1 | YES | YES | YES |
| CHEM_07 | Pressure | 4685 | 1 | Open-loop data; stiction | 1–4685 | 1 | YES | NO | NO |
| CHEM_08 | Pressure | 900 | 1 | Open-loop data; stiction | 1–900 | 1 | YES | NO | NO |
| CHEM_09 | Pressure | 2732 | 1 | Stiction | 1–2732 | 1 | YES | YES | YES |
| CHEM_10 | Pressure | 1000 | 1 | Stiction | 1–1000 | 1 | YES | YES | YES |
| CHEM_11 | Flow | 1000 | 1 | Stiction | 1–1000 | 1 | YES | YES | YES |
| CHEM_12 | Flow | 2000 | 1 | Stiction | 1–2000 | 2 | YES | NO | YES |
| CHEM_13 | Analyser | 1500 | 20 | Faulty steam sensor; no stiction | 1–1500 | 1 | NO | NO | NO |
| CHEM_14 | Flow | 1500 | 20 | Faulty steam sensor; no stiction | 1–1500 | 1 | NO | NO | NO |
| CHEM_15 | Pressure | 1500 | 20 | Interaction (likely); no stiction | 1–1500 | 1 | NO | NO | NO |
| CHEM_16 | Pressure | 1500 | 20 | Interaction (likely); no stiction | 1–1500 | 1 | NO | NO | NO |
| CHEM_17 | Temperature | 1500 | 20 | Faulty steam sensor; no stiction | 1–1500 | 1 | NO | NO | NO |
| CHEM_18 | Flow | 1040 | 12 | Stiction (likely) | 1–1040 | 1 | YES | YES | YES |
| CHEM_19 | Flow | 721 | 12 | Stiction (likely) | 1–721 | 1 | YES | NO | NO |
| CHEM_20 | Flow | 721 | 12 | Stiction (likely) | 1–721 | 1 | YES | NO | NO |
| CHEM_21 | Flow | 721 | 12 | Disturbance (likely) | 1–721 | 1 | NO | NO | NO |
| CHEM_22 | Flow | 721 | 12 | Stiction (likely) | 1–721 | 1 | YES | YES | YES |
| CHEM_23 | Flow | 1500 | 12 | Stiction (likely) | 1–1500 | 1 | YES | YES | YES |
| CHEM_24 | Flow | 1500 | 12 | Stiction (likely) | 1–1500 | 0.5 | YES | YES | YES |
| CHEM_25 | Pressure | 721 | 12 | Stiction + possible margin stability | 1–721 | 0.5 | NO | NO | YES |
| CHEM_26 | Level | 1094 | 12 | Stiction (likely) | 1–1094 | 1 | YES | YES | YES |
| CHEM_27 | Level | 1333 | 12 | Disturbance (likely) | 1–1333 | 1 | NO | YES | YES |
| CHEM_28 | Temperature | 721 | 12 | Stiction (likely) | 1–721 | 1 | YES | YES | YES |
| CHEM_29 | Flow | 7201 | 60 | Stiction | 1–4096 | 1 | YES | NO | NO |
| CHEM_30 | Flow | 17281 | 15 | Stiction | 1–4096 | 1 | YES | YES | YES |
| CHEM_32 | Flow | 1998 | 10 | Stiction (likely) | 1–1998 | 1 | YES | YES | YES |
| CHEM_33 | Flow | 721 | 12 | Disturbance (likely) | 1–721 | 1 | NO | NO | NO |
| CHEM_34 | Flow | 719 | 10 | Disturbance (likely) | 1–719 | 1 | NO | NO | NO |
| CHEM_35 | Flow | 2000 | 10 | Stiction (likely) | 1–2000 | 1 | YES | YES | YES |
| CHEM_36 | Level | 804 | 12 | Disturbance (likely) | 1–804 | 1 | NO | NO | NO |
| CHEM_37 | Level | 1711 | 12 | Disturbance (likely) | 1–1711 | 1 | NO | NO | NO |
| CHEM_38 | Pressure | 933 | 10 | Disturbance (likely) | 1–933 | 1 | NO | NO | NO |
| CHEM_39 | Pressure | 719 | 60 | Disturbance (likely) | 1–719 | 1 | NO | NO | NO |
| CHEM_40 | Temperature | 1441 | 60 | No clear oscillation (according to power spectrum) | 101–1124 | 1 | NO | NO | NO |
| CHEM_41 | Temperature | 1441 | 60 | OP saturation | 1–1441 | 1 | NO | NO | NO |
| CHEM_44 | Temperature | 1441 | 60 | Too few cycles; no clear oscillation; OP saturation | 1–1441 | 2 | NO | NO | NO |
| CHEM_45 | Pressure | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_46 | Pressure | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_47 | Pressure | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_48 | Pressure | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_52 | Level | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_53 | Level | 1441 | 60 | No clear oscillation | 1–1441 | 1 | NO | NO | NO |
| CHEM_54 | Level | 1441 | 60 | No clear oscillation | 1–1441 | 1 | NO | NO | NO |
| CHEM_56 | Flow | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_58 | Flow | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_59 | Flow | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_61 | Flow | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| CHEM_62 | Flow | 1441 | 60 | No clear oscillation (according to power spectrum) | 1–1441 | 1 | NO | NO | NO |
| MET_01 | Gauge | 1716 | 0.05 | External disturbance (likely) | 1–1716 | 1 | NO | NO | NO |
| MET_02 | Gauge | 4411 | 0.05 | External disturbance (likely) | 1–4411 | 1 | NO | NO | NO |
| MET_03 | Gauge | 5642 | 0.05 | No oscillation | 1–5642 | 1 | NO | NO | NO |
| MIN_01 | Temperature | 2641 | 60 | Stiction | 1–2641 | 1 | YES | YES | YES |
| PAP_01 | Flow | 849 | 1 | Stiction | 1–849 | 1 | YES | NO | NO |
| PAP_02 | Flow | 1196 | 1 | Stiction | 1–1196 | 1 | YES | YES | YES |
| PAP_03 | Level | 1147 | 1 | Stiction | 1–1147 | 1 | YES | YES | YES |
| PAP_04 | Concentration | 1196 | 1 | Deadzone and tight tuning; no stiction | 1–1196 | 1 | NO | YES | YES |
| PAP_05 | Concentration | 18000 | 0.2 | Stiction | 1–4096 | 1 | YES | YES | YES |
| PAP_06 | Level | 846 | 1 | No stiction | 1–846 | 1 | NO | NO | NO |
| PAP_07 | Flow | 14101 | 0.2 | External disturbance | 5–4096 | 1 | NO | NO | NO |

(continued on next page)

**Table 11** (*continued*).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PAP_08 | Level | 1800 | 5 | No stiction | 1–1800 | 1 | NO | NO | NO |
| PAP_09 | Temperature | 1800 | 5 | No stiction | 1–1800 | 1 | NO | YES | YES |
| PAP_11 | Level | 4179 | 15 | Stiction | 1–4179 | 1 | YES | NO | NO |
| PAP_12 | Level | 4462 | 15 | Stiction | 1–4462 | 1 | YES | YES | YES |
| PAP_13 | Level | 4237 | 15 | Stiction | 1–4237 | 1 | YES | YES | YES |
| POW_01 | Level | 8641 | 5 | Stiction | 1–4096 | 1 | YES | NO | NO |
| POW_02 | Level | 8641 | 5 | Stiction | 1–4096 | 1 | YES | YES | YES |
| POW_03 | Level | 8641 | 5 | No stiction | 1–4096 | 1 | NO | NO | NO |
| POW_04 | Level | 8641 | 5 | Stiction | 1–4096 | 1 | YES | YES | YES |
| POW_05 | Level | 8641 | 5 | No stiction | 1–4096 | 1 | NO | NO | NO |

it is not clear whether the method or the data has led to the increase in accuracy over other methods.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Harrison ONeill reports financial support was provided by European Regional Development Fund. Harrison ONeill reports financial support was provided by Spiro Control Ltd.

## Acknowledgements

The authors would like to thank the European Regional Development Fund for sponsoring this research as part of the Eco-Innovation Cheshire and Warrington project (03R17P01835). Thanks also to Spiro Control Ltd. for co-sponsoring and supervising this work. Finally, thanks to Jelali and Huang (and all other contributors) for providing the industrial benchmark data for testing as well as Chris Catterall, Senior Advanced Process Controls Consultant at Phillips 66 Limited for supplying new control loop data for additional testing.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.dche.2023.100116.

## References

Amiruddin, A.A.A.M., Zabiri, H., Jeremiah, S.S., Teh, W.K., Kamaruddin, B., 2019. Valve stiction detection through improved pattern recognition using neural networks. Control Eng. Pract. 90, 63–84. http://dx.doi.org/10.1016/j.conengprac.2019.06.008, URL http://www.sciencedirect.com/science/article/pii/S0967066119300838.

Åström, K.J., Wittenmark, B., 1997. Computer-Controlled Systems, third ed. Prentice-Hall, Inc., USA.

Bacci di Capaci, R., Scali, C., 2018. Review and comparison of techniques of analysis of valve stiction: From modeling to smart diagnosis. Chem. Eng. Res. Des. 130, 230–265. http://dx.doi.org/10.1016/j.cherd.2017.12.038, URL http://www.sciencedirect.com/science/article/pii/S0263876217307116.

Bauer, M., Auret, L., Bacci di Capaci, R., Horch, A., Thornhill, N.F., 2020. Pid data, https://sacac.org.za/resources/. (Accessed 27 May 2020).

Bauer, M., Horch, A., Xie, L., Jelali, M., Thornhill, N., 2016. The current state of control loop performance monitoring - a survey of application in industry. J. Process Control 38, 1–10. http://dx.doi.org/10.1016/j.jprocont.2015.11.002, URL http://www.sciencedirect.com/science/article/pii/S0959152415002127.

Choudhury, M.S., Shah, S., Thornhill, N., 2004. Diagnosis of poor control-loop performance using higher-order statistics. Automatica 40 (10), 1719–1728. http://dx.doi.org/10.1016/j.automatica.2004.03.022, URL http://www.sciencedirect.com/science/article/pii/S0005109804001499.

Choudhury, M.S., Thornhill, N., Shah, S., 2005. Modelling valve stiction. Control Eng. Pract. 13 (5), 641–658. http://dx.doi.org/10.1016/j.conengprac.2004.05.005, URL http://www.sciencedirect.com/science/article/pii/S0967066104001145.

Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W., 2018. Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh a Python package). Neurocomputing 307, 72–77. http://dx.doi.org/10.1016/j.neucom.2018.03.067, URL http://www.sciencedirect.com/science/article/pii/S0925231218304843.

Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W., 2020. Overview of extracted features. https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html. (Accessed 04 September 2020).

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20 (3), 273297. http://dx.doi.org/10.1023/A:1022627411411.

Damarla, S.K., Sun, X., Xu, F., Shah, A., Huang, B., 2022. A sigmoid function based method for detection of stiction in control valves. In: 2022 IEEE International Symposium on Advanced Control of Industrial Processes. AdCONIP, pp. 210–215. http://dx.doi.org/10.1109/AdCONIP55568.2022.9894234.

Dambros, J.W.V., Farenzena, M., Trierweiler, J.O., 2016. Data-based method to diagnose valve stiction with variable reference signal. Ind. Eng. Chem. Res. 55 (39), 10316–10327. http://dx.doi.org/10.1021/acs.iecr.6b01234.

Farenzena, M., Trierweiler, J.O., 2009. A novel technique to estimate valve stiction based on pattern recognition. In: de Brito Alves, R.M., do Nascimento, C.A.O., Biscaia, E.C. (Eds.), 10th International Symposium on Process Systems Engineering: Part A. In: Computer Aided Chemical Engineering, vol. 27, Elsevier, pp. 1191–1196. http://dx.doi.org/10.1016/S1570-7946(09)70589-9, URL http://www.sciencedirect.com/science/article/pii/S1570794609705899.

Forsman, K., Stattin, A., 1999. A new criterion for detecting oscillations in control loops. In: 1999 European Control Conference. ECC, pp. 2313–2316. http://dx.doi.org/10.23919/ECC.1999.7099666, URL https://ieeexplore.ieee.org/abstract/document/7099666.

Garcia, C., 2008. Comparison of friction models applied to a control valve. Control Eng. Pract. 16 (10), 1231–1243. http://dx.doi.org/10.1016/j.conengprac.2008.01.010, URL http://www.sciencedirect.com/science/article/pii/S0967066108000221.

He, Q.P., Wang, J., Pottmann, M., Qin, S.J., 2007. A curve fitting method for detecting valve stiction in oscillating control loops. Ind. Eng. Chem. Res. 46 (13), 4549–4560. http://dx.doi.org/10.1021/ie061219a.

Henry, Y., Aldrich, C., Zabiri, H., 2020. Detection and severity identification of control valve stiction in industrial loops using integrated partially retrained CNN-PCA frameworks. Chemometr. Intell. Lab. Syst. 206, 104143. http://dx.doi.org/10.1016/j.chemolab.2020.104143, URL http://www.sciencedirect.com/science/article/pii/S0169743920301076.

Horch, A., 1999. A simple method for detection of stiction in control valves. Control Eng. Pract. 7 (10), 1221–1231. http://dx.doi.org/10.1016/S0967-0661(99)00100-8, URL http://www.sciencedirect.com/science/article/pii/S0967066199001008.

Horch, A., 2010. In: Jelali, M., Huang, B. (Eds.), Detection and Diagnosis of Stiction in Control Loops: State of the Art and Advanced Methods. Springer London, London, pp. 132–147. http://dx.doi.org/10.1007/978-1-84882-775-2_13.

Hotelling, H., 1933. Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. 24 (6), 417–441. http://dx.doi.org/10.1037/h0071325.

Huang, B., 2020. International stiction database, https://sites.ualberta.ca/~bhuang/Stiction-Book.htm. (Accessed 27 May 2020).

Jelali, M., Huang, B., 2010. Detection and Diagnosis of Stiction in Control Loops: State of the Art and Advanced Methods. Springer London, http://dx.doi.org/10.1007/978-1-84882-775-2.

Kamaruddin, B., Zabiri, H., Amiruddin, A.M., Teh, W., Ramasamy, M., Jeremiah, S., 2020. A simple model-free butterfly shape-based detection (BSD) method integrated with deep learning CNN for valve stiction detection and quantification. J. Process Control 87, 1–16. http://dx.doi.org/10.1016/j.jprocont.2020.01.001, URL http://www.sciencedirect.com/science/article/pii/S0959152419304639.

Kano, M., Maruta, H., Kugemoto, H., Shimizu, K., 2004. Practical model and detection algorithm for valve stiction. IFAC Proc. Vol. 37 (9), 859–864. http://dx.doi.org/10.1016/S1474-6670(17)31917-1, URL http://www.sciencedirect.com/science/article/pii/S1474667017319171, 7th IFAC Symposium on Dynamics and Control of Process Systems 2004 (DYCOPS -7), Cambridge, USA, 5-7 July, 2004.

Karra, S., Karim, M.N., 2009. Comprehensive methodology for detection and diagnosis of oscillatory control loops. Control Eng. Pract. 17 (8), 939–956. http://dx.doi.org/10.1016/j.conengprac.2009.02.015, URL http://www.sciencedirect.com/science/article/pii/S0967066109000392.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105, URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kumari, P., Halim, S.Z., Kwon, J.S.I., Quddus, N., 2022a. An integrated risk prediction model for corrosion-induced pipeline incidents using artificial neural network and Bayesian analysis. Process Saf. Environ. Prot. 167, 34–44. http://dx.doi.org/10.1016/j.psep.2022.07.053, URL https://www.sciencedirect.com/science/article/pii/S0957582022006656.

Kumari, P., Wang, Q., Khan, F., Kwon, J.S.I., 2022b. A direct transfer entropy-based multiblock Bayesian network for root cause diagnosis of process faults. Ind. Eng. Chem. Res. 61 (43), 16166–16178. http://dx.doi.org/10.1021/acs.iecr.2c02320.

Lang, X., Lu, S., Xie, L., Zakharov, A., Zhong, D., Jms-Jounela, S.L., 2018. Bihocerence based industrial control loop nonlinearity detection and diagnosis in short nonstationary time series. J. Process Control 63, 15–28. http://dx.doi.org/10.1016/j.jprocont.2018.01.001, URL http://www.sciencedirect.com/science/article/pii/S0959152418300015.

Lee, K.H., Ren, Z., Huang, B., 2008. Novel closed-loop stiction detection and quantification method via system identification.

Mahadevan, S., Shah, S.L., 2009. Fault detection and diagnosis in process data using one-class support vector machines. J. Process Control 19 (10), 1627–1639. http://dx.doi.org/10.1016/j.jprocont.2009.07.011, URL http://www.sciencedirect.com/science/article/pii/S0959152409001334.

Pechenizkiy, M., Tsymbal, A., Puuronen, S., 2006. On combining principal components with Fisher's linear discriminants for supervised learning. Found. Comput. Decis. Sci. 31 (1), 59–73.

Rawat, W., Wang, Z., 2017. Deep convolutional neural networks for image classification: A comprehensive review. Neural Comput. 29 (9), 23522449. http://dx.doi.org/10.1162/neco_a_00990.

Rossi, M., Scali, C., 2005. A comparison of techniques for automatic detection of stiction: simulation and application to industrial data. J. Process Control 15 (5), 505–514. http://dx.doi.org/10.1016/j.jprocont.2004.11.003, URL http://www.sciencedirect.com/science/article/pii/S0959152404001192.

Scali, C., Ghelardoni, C., 2008. An improved qualitative shape analysis technique for automatic detection of valve stiction in flow control loops. Control Eng. Pract. 16 (12), 1501–1508. http://dx.doi.org/10.1016/j.conengprac.2008.04.009, URL http://www.sciencedirect.com/science/article/pii/S0967066108000798.

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J., 2000. Support vector method for novelty detection. In: Advances in Neural Information Processing Systems 12. MIT Press,Max-Planck-Gesellschaft, Cambridge, MA, USA, pp. 582–588.

Singhal, A., Salsbury, T.I., 2005. A simple method for detecting valve stiction in oscillating control loops. J. Process Control 15 (4), 371–382. http://dx.doi.org/10.1016/j.jprocont.2004.10.001, URL http://www.sciencedirect.com/science/article/pii/S0959152404001106.

Teh, W.K., Zabiri, H., Samyudia, Y., Jeremiah, S.S., Kamaruddin, B., Mohd Amiruddin, A.A.A., M. Ramli, N., 2018. An improved diagnostic tool for control valve stiction based on nonlinear principle component analysis. Ind. Eng. Chem. Res. 57 (33), 11350–11365. http://dx.doi.org/10.1021/acs.iecr.8b01012.

Thornhill, N., Huang, B., Zhang, H., 2003. Detection of multiple oscillations in control loops. J. Process Control 13 (1), 91–100. http://dx.doi.org/10.1016/S0959-1524(02)00007-0, URL http://www.sciencedirect.com/science/article/pii/S0959152402000070.

Van Rossum, G., 2020. The Python Library Reference, Release 3.8.3. Python Software Foundation, URL https://docs.python.org/3/library/pickle.html.

Xie, L., Cong, Y., Horch, A., 2013. An improved valve stiction simulation model based on ISA standard tests. Control Eng. Pract. 21 (10), 1359–1368. http://dx.doi.org/10.1016/j.conengprac.2013.06.013, URL http://www.sciencedirect.com/science/article/pii/S0967066113001172.

Zabiri, H., Ramasamy, M., 2009. NLPCA as a diagnostic tool for control valve stiction. J. Process Control 19 (8), 1368–1376. http://dx.doi.org/10.1016/j.jprocont.2009.04.010, URL http://www.sciencedirect.com/science/article/pii/S0959152409000821, Special Section on Hybrid Systems: Modeling, Simulation and Optimization.

Zhang, K., Liu, Y., Gu, Y., Ruan, X., Wang, J., 2022. Multiple-timescale feature learning strategy for valve stiction detection based on convolutional neural network. IEEE/ASME Trans. Mechatronics 27 (3), 1478–1488. http://dx.doi.org/10.1109/TMECH.2021.3087503.

Zheng, D., Sun, X., Damarla, S.K., Shah, A., Amalraj, J., Huang, B., 2021. Valve stiction detection and quantification using a K-means clustering based moving window approach. Ind. Eng. Chem. Res. 60 (6), 2563–2577. http://dx.doi.org/10.1021/acs.iecr.0c05609.