Analysis of techniques for automatic detection and quantification of stiction in control loops[1]

Henrik Manum

26th July 2006

[1]Diploma work spring 2006 (NTNU).

Abstract

This thesis focuses on detection and quantification of stiction in control valves. Emphasis is on application of the Yamashita stiction detection method [34], which is a method based on qualtitative description formalism. Basic properties and main factors are put into evidence by application on data generated by simulation, while the reliability is checked by application on plant data sets to account for sensitivity to noise, for the effect of set point variations due to cascade or advanced control acting on the upper level. The algorithm shows to be able to detect stiction when it shows up with clear patterns (about 50% of examined cases coming from about 200 data sets). This allows a quicker detection and saving of computation time with respect to more comprehensive techniques, thus suggesting on line implementation of the technique. For quantification, only introductory work is conducted. A method for fitting an ellipse to noisy industrial data by first filtering the data with an approximate Wiener filter is studied.

Keywords: Stiction, Pattern recognition, PID controller, FFT, Yamashita

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology.

27. July 2006

Henrik Manum

Acknowledgments

I want to thank prof. Claudio Scali for accepting me as his student and prof. Sigurd Skogestad for allowing me to write my thesis at the University of Pisa. The spring of 2006 has been excellent. While staying in Pisa and living with Italians on a daily basis I feel I have gotten insight into a culture rather different from the one back home. The way almost all people have been open and friendly to me has made the stay very pleasant.

Vorrei ringraziare tutti gli studenti a Pisa. Voi siete stati molto molto gentili con me. Sono sicuro che mi ricorderò questi mesi finché vivrò.

I dedicate this thesis to the Norwegian ski-jumper Tommy Ingebrigtsen.

Contents

Chapter 1

Introduction

It is a sad fact that many control loops actually degrades performance, by increasing the variability rather than decreasing it [10]. There are many possible causes for degrading performance, some are listed in [10]:

- Operating conditions have changed after the controller was tuned.
- The actual process has changed, some process modifications have been made after the controller was tuned.
- The controller still uses the default settings, it has never been tuned.
- A poor (or even inconsistent) control structure with significant interactions between the loops.
- Some equipment in the control loop may be in need of maintenance or replacement, e.g.
  - faulty measurements
  - control valves with stiction
  - fouling in heat exchangers
  - etc.

High performance of the control loops is necessary to ensure high product quality and low cost in chemical plants. Often poor performance can be detected by the operator, but many times the problem may propagate to other loops, making it difficult to detect the root cause [7]. This work will focus on one of the many reasons for degrading performance listed above, namely stiction and its automatic detection. Among the many types of nonlinearities in control valves, stiction is the most commonly occurring phenomenon and one of the long-standing problems in process industry [4].

1.1 Outline of this thesis

This work was conducted at the Chemical Process Control Laboratory (CPC-Lab) at the University of Pisa, Italy. The supervisor has been prof. Scali. The work of prof. Scali and his students has in the recent years focused on plant monitoring, with emphasis on detecting and characterizing oscillating loops. A software package called "Plant Check-Up" (PCU) has been developed. The architecture of this package will briefly be reviewed in this introductory chapter. Some years ago the main focus was on auto-tuning. Today detection of stiction is more "hot". The goal is to find/develop more routines for stiction detection and quantification to be added to the software package.

This chapter will give an introduction to stiction and its properties and how the properties are used to develop automatic routines to detect it. A review of some popular methods for stiction detection will also be conducted.

A major part of the work conducted at Pisa consisted in reviewing and testing a rather new method for stiction detection proposed by Yamashita [34]. (From now on the Yamashita stiction detection method will simply be referred to as Yam). An introduction to this method will be included in this chapter.

Chapter 2 will be a report of work conducted by applying the Yam method on both simulated and real plant data. The work resulted in a conference paper at the ANIPLA conference in Rome, November, 2006[1]. Chapter 2 will be a larger version of that paper. Most of my time here at Pisa was used to do the work that resulted in that paper and also writing the paper itself.

The Yam method is based on pattern recognition of plotting the process variable versus the control signal. While working with the method on industrial data some patterns not reported in the investigated literature was found, and they will be discussed in chapter 3.

After stiction has been detected by an appropriate system it is important to quantify the presence of stiction to select the loops that are in the highest need of maintenance [4]. This can be done using a plot of the process variable versus the controller output, and chapter 4 will give an introduction to this problem and report some work conducted. Unfortunately this work is not finished as time ran out.

Chapter 5 will give a summary of the thesis and suggestions to further work. The thesis will also contain some appendices with Matlab scripts and a CD with implemented software, and finally (appendix E) a copy of the paper submitted to ANIPLA.

[1] See www.anipla.it/anipla2006.

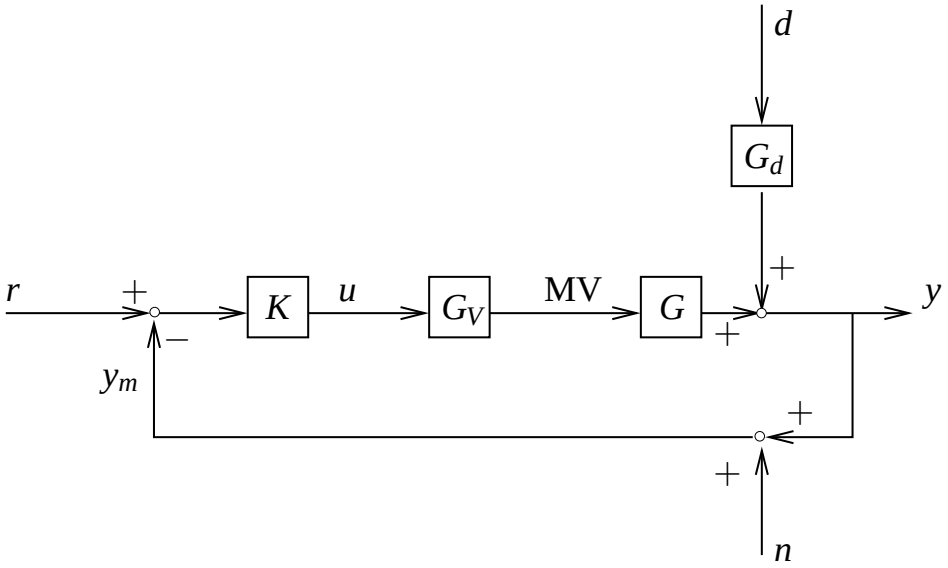

Figure 1.1: Feedback loop with definition of variables. Inspired by [26].

1.2 Notation

This section will give an overview of the notation used in this thesis.

Figure 1.1 shows an usual feedback loop with a transfer function G_V representing the valve between the controller and the process. The loop receives the command signal r. This signal will often be denoted as SP, but also y_{sp}. The error signal is the difference between the measured process variable PV (or y, or y_m) and the reference r, that is $e = r - y_m$. This signal is fed to the controller (K or G_C), which calculates the control input u. Often OP will be used in place of u. Since valves are important for this work, the control signal u is sent to the valve transfer function G_v. The output from the valve is always denoted MV, and is the input to the process G, which has the output signal y or PV. Disturbances can be entered though the disturbance transfer function G_d by using the signal d.

In the model noise can be added to y, giving $y_m = y + n$. In chapter 2 there will also be noise added to the measurements of OP and MV.

This general one-degree-of-freedom control configuration in taken from [26], though with an extra transfer function to represent the valve.

In this thesis the output signal from the valve, MV, and the output from the process, PV, will be used interchangeably. This is because mostly flow-loops with incompressible fluids are considered. Usually one can assume that the dynamics between the valve position and the flow rate are very fast, thus PV can be assumed proportional to MV. This is will be more clear after the Yam method is described.

For vectors (typically signals/measured variables) the following notation will be used: $x = (x_1, x_2, \ldots, x_N) = [x_1 \quad x_2 \quad \ldots \quad x_N]^T = (x_j)_{j=1}^N$. Here, $x \in \mathscr{C}^N$ (often restriced to $\mathscr{R}^N$), and $x_j \in \mathscr{C}$ (often restriced to $\mathscr{R}$) for $j = 1, \ldots, N$. Representing the vector x as a sequence $(x_j)_{j=1}^N$ may be a bit inaccurate as really the space in which the sequence lies in is different from the vector space $\mathscr{C}^N$. For now I state that the squencies are bounded, i.e. $(x_j)_{j=1}^N$such that $|x_j| < \varepsilon \in \mathscr{C}$. Adding more formalism should not be necassary for the simple use in this thesis.

1.3 An introduction to stiction

As the main focus of this thesis is automatic detection (and quantification) of stiction, an introduction to this phenomenon will now be given.

On Wikipedia[2] they claim that stiction is a informal contraction of the term "static friction", perhaps also influenced by the verb "stick". They further give a nice analogy to car driving, for which I feel at home:

It is well known from physics that the static friction is larger than the dynamic friction. Growing up in Norway, I have often experienced the difference between static and dynamic friction while driving the car. When the wheel is rolling there is static friction between the wheel and the surface (in Norway at wintertime typically ice or snow), whereas between a sliding or spinning wheel and the surface there is dynamic friction. The static friction may be seen upon as a threshold at which a rolling object would begin to slide over a surface rather than rolling at the expected rate (or spin at a too high rate).

Taking the "slippery-surface driving course", which is compulsory in Norway, I learned that when the car is sliding I should release the brakes and steer the wheels in the direction of the vector of the movement. This way the wheels get a chance to regain the static friction, and the control over the car is improved. This maneuver is called counter-steering. Anti-lock braking systems should release the brakes automatically when the wheels are sliding, and other systems such as automatic stabilization control try to reduce the force from the engine when the wheels start spinning, ususally with the goal of regaining the static friction.

Figure 1.2 shows a typical expected plot of a loop in a chemical plant suffering from stiction in valve output (manipulated variable, MV) versus valve input (controller output, OP). The figure is taken from [5], which gives a good introduction to the phenomenon. In simple words, what can happen in a loop affected by stiction is this:

- At time $t < 0$ the loop is at rest, with the error signal $e(t) = 0$.

[2] www.wikipedia.co.uk

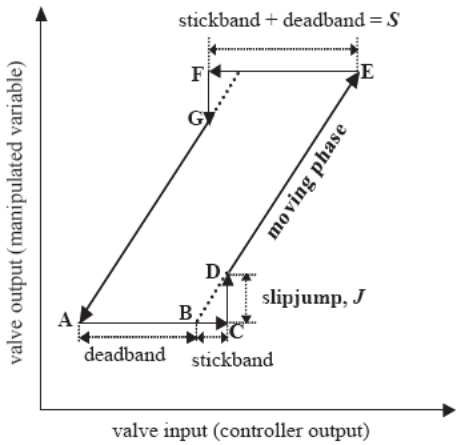

Figure 1.2: Typical stiction pattern. Taken from [5].

- At time $t = 0$ either a disturbance or change in the command signal occurs, causing the error signal to deviate from 0.
- Since the valve is stuck, the error will remain non-zero.
- For a controller with integral action the controller output will increase as long as the error is non-zero and has the same sign.
- When the force caused by the controller output on the valve is large enough to overcome the static friction the valve will jump, causing a slip-jump. In this slip-jump energy stored in the valve system is released, thus there is an extra push on the valve.
- Now the valve is under dynamic friction and is moving as expected.
- If the velocity of the valve should decrease below a given thershold, or equally if the derivative of the OP should be too low, the valve will stick again.

In figure 1.2 the “sticky period” when the valve is not moving is composed by the deadband and the stickband. If the valve stops and sticks and then starts again while traveling in the same direction only the stickband needs to be overcome. In practice it is difficult to observe or measure the stickband, it is therefore more reasonable to add these two terms together to make the variable S. The size of the slip-jump will from now on be called J. Also the size of the slip-jump is difficult to observe in plant data, but for conducting simulations the parameter is needed, as it represents the abrupt release of energy when the valve jumps.

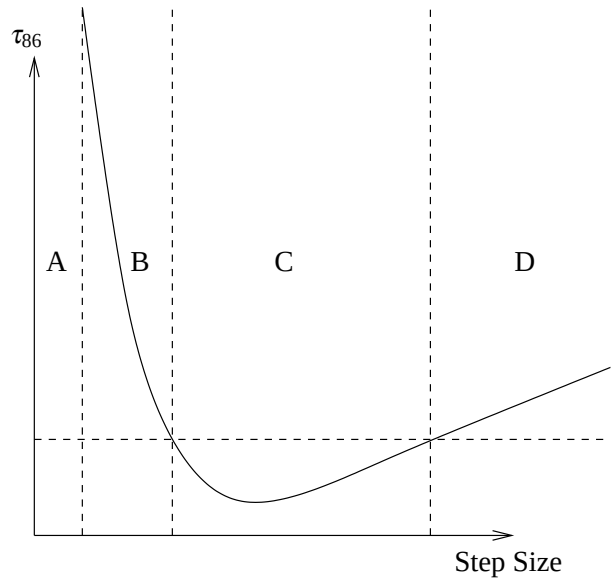

Figure 1.3: Illustrations of different regions in response time to step changes as defined in [1]. The horizontal dotted line is some user-specification on maximum allowed τ_{86}.

Let us summarize the phenomenon by the definition proposed in [5]:

Stiction Stiction is a property of an element such that its smooth movement in response to a varying input is preceded by a sudden abrupt jump called the slip-jump. Slip-jump is expressed as a percentage of the output span. Its origin in a mechanical system is static friction which exceeds the friction during smooth movement.

EnTech, which is an engineering company specialized in the optimization of process performance, particularly in the pulp and paper industry, has given some "industrial" definitions of different operating regions for valves suffering from non-linearities [1], which are rather good to achieve insight, and will now be recaptured here.

For a pneumatically actuated control valve, there are four regions of nonlinear operation (referred to as regions A, B, C and D). The regions are illustrated in figure 1.3. For a very small input signal step change (say 0.1%) the closure member does not move at all in a reasonable time after the step change. This is region A, and this region is less than dead band or step resolution. Above some initial threshold (say 0.1% to 1%) motion occurs, but due to nonlinearities and other effects the responses are not consistent. This is known as region B. For larger step sizes the closure member moves in a more consistent manner, an it is possible to

classify responses in this region (Region C) such that their step response times (typically some specification on the response in time domain) all fall below the acceptable limit which is required for effective control. For step changes which are larger still (Region D), it's likely that the motion of the control valve system will become velocity limited (steps of say 10% or greater), hence causing the step response times to be progressively longer and too slow for acceptable control.

As disturbances occur the controller will first take small corrective actions, causing the valve to stay in region A. The loop is now essentially open, as the valve does not move. It is likely that the controller now "winds up" and moves the valve into region B. Here the output is inconsistent with possibly significantly variable (long) delays. For feedback control to work the controller need to penetrate into region C to give reliable responses[3]. For large disturbances the control output can force the valve system into region D, where the response is too slow to effective control. This means that the effective time-constant of the process is larger than what the controller was tuned for, and hence instabilities may occur. (The controller is tuned too tight for these "unexpected" large disturbances.)

As a summary, the industrially defined operating regions for the valve are defined directly to assess the performance of the valve system. The sizes of regions A and B can be affected by the presence of stiction. In region A the valve is sticky and hence not moving. In region B the uncertain gain in the valve can be explained by the abrupt release of energy when the valve slips.

1.4 Stiction modeling

Some of the work in this thesis has been done utilizing a data-driven model of a valve. This model was developed by Choudhury et al. [5]. The reason for using this model and not a first principles/ physical model is its simplicity. To further justify the choice of using a data-driven model a short review of a physically based model will now be given, thereafter the data-driven model will be discussed.

1.4.1 Physical model

The reference for this subsection is [5].

For a pneumatic sliding stem valve the force balance equation based on Newton's second law can be written

$$M\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = \sum_{\text{Forces}} = F_a + F_r + F_f + F_p + F_{is}, \tag{1.1}$$

[3]Hence large gain feedback is favorable to push the system into region C.

where M is the mass of the moving parts, $F_a = Au$ is the force applied by the pneumatic actuator where A is the area of the diaphragm and u is the actuator air pressure, $F_r = -kx$ is the spring force with spring constant k, $F_p = -\alpha\Delta P$ is the force due to fluid pressure drop where α is the plug unbalance area and ΔP is the fluid pressure drop across the valve, F_i is the extra force required to force the valve to be into the seat and F_j is the friction force.

For now the two forces F_i and F_p will be assumed to be negligible.

A model for the friction term F_f can be

$$F_f = \begin{cases} -F_c\text{sign}(v) - vF_v & \text{if } v \neq 0, \\ -(F_a + F_r) & \text{if } v = 0 \text{ and } |F_a + F_r| \leq F_s, \\ -F_s\text{sign}(F_a + F_r) & \text{if } v = 0 \text{ and } |F_a + F_r| > F_s. \end{cases} \tag{1.2}$$

The first case, when the valve is moving, $v \neq 0$, is compounded of a velocity independent term F_c, known as Coulomb friction, and a viscous friction term vF_v that is a linear function of the velocity.

When the valve gets stuck the valve is not moving as long as the applied forces F_a and F_r are less than a threshold F_s. This threshold is known as the maximum static friction. (One sees this directly from the force balance when the valve is not moving with zero acceleration, $\frac{d^2x}{dt^2} = 0$).

At the time when the applied forces F_a and F_r overcomes the maximum static friction for the system the valve starts to move. At this instant the sum of forces is $(F_a + F_r) - F_s\text{sign}(F_a + F_r)$ and the valve starts to accelerate.

The obvious disadvantage with applying the model presented above to a generic valve is the need to specify a rather large set of parameters. Parameters that need to be specified are M, F_s, F_c, F_v, k, A, and air pressure, which are 7 parameters.

To draw the analogy back to car-driving, F_s is the threshold of the force between the tires and the tarmac (or snow/ice) to make the wheels spin or slide.

1.4.2 A data-driven model

A data-driven model that has been used at the lab before I came here is a model developed by Choudhury [5]. A flow-sheet for the model can be found in the reference. The model consists of the two parameters defined graphically in figure 1.2, namely the size of deadband plus stickband S and the size of slip jump J. In [5] it is shown that the model works good in comparison to the physical model shown above, and this is the main reason for why this model is used in the lab.

The implementation of the model used in the lab was given to us from the author of [5]. Reading the paper ([5]) carefully, it should be noted that “In order to handle stochastic inputs, the model requires the implementation of a PI(D) controller including its filter under a full industrial specification environment”.

This implies that adding noise to the model without the filter on the input to the valve may be problematic. This is further discussed in chapter 2, where the model is used to generate simulated data for valves with and without stiction, and noise was introduced.

It should also be noted that is it possible to make a very simple model of stiction if one disregards the slip-jump J. The model is then to hold the valve when it comes to rest until the control signal has changed enough to overcome some given threshold. After that the valve is moving freely, without any initial push from the slip-jump. Remember that the slip-jump originates from the abrupt release of static energy when the static friction is overcome. Of course this "stored" energy must convert into another form of energy, in this case the extra push in from of a slip-jump. However, in simulation, just by releasing the valve after it has being stuck gives a similar effect, because the control signal can be very high and the valve is "pushed" for this reason. The Choudhury model was used for most of this work, but I want to point out that an even simpler model than the two-parameter model of [5] can be used to get insight into some of the basic features of a sticky valve.

1.5 Automatic detection of stiction

Several techniques for automatic detection of stiction in control loops have been proposed in literature. A brief review of three popular methods will now be given.

A good stiction detection method should manage to identify stiction correctly and not wrongly characterize for instance changes is the set point or disturbances as stiction in the loops. It is desired that the methods has some index that gives a measure of how much stiction is present. Further it is desired that the methods does not have any uncertainty regions. If this is the case they should be well defined and preferably as small as possible.

1.5.1 Cross-correlation technique

Horch [9] has proposed a method based on the cross-correlation between the control signal and the process output. This method is often used as a reference in other articles for stiction detection. A common factor in those articles is that the other methods are better than the method by Horch because it often gives wrong indications. See for example [34] or [12]. Even though it is shown to often give wrong decisions it is also regarded as being popular and easy to implement [20].

The following assumptions needs to be valid for application of the cross-correlation method:

- The process does not have an integral action.

- The process is controlled by a PI-controller.
- The oscillating loop has been detected as being oscillating with a significantly large amplitude.

For a data set with N data points, the cross-correlation function between u and y for lag τ (where τ is an integer) is given by [10]:

$$r_{uy}(\tau) = \frac{\sum_{k=k_0}^{k_1} u(k)y(k+\tau)}{\sum_{k=1}^{N} u(k)y(k)}. \tag{1.3}$$

where

$$\begin{aligned} k_0 &= 1 \quad \text{for} \quad \tau \geq 0 \\ k_0 &= \tau + 1 \quad \text{for} \quad \tau < 0 \\ k_1 &= N - \tau \quad \text{for} \quad \tau \geq 0 \\ k_1 &= N \quad \text{for} \quad \tau < 0. \end{aligned}$$

For an external oscillating disturbance the phase lag in the cross-correlation is $-\pi$, while it is $-\pi/2$ for a loop with stiction present.

Using the method two parameters are calculated. According to the values of these parameters the loop will be regarded as either sticky or subject to sinusoidal perturbations. There is also an uncertainty region where no decision can be taken [20].

1.5.2 Bi-coherence method

Choudhury et al. [4] have proposed a method based on Higher Order Statistics. It is observed that the first and second order statics (mean, variance, autocorrelation, power spectrum etc.) are only sufficient to describe linear systems. Non-linear behavior must be detected using higher order statics such as "bi-spectrum" and "bi-coherence".

References [6, 7] gives a short introduction to higher order statistics. The bi-spectrum is defined as

$$B(f_1, f_2) \triangleq E\left[X(f_1)X(f_2)X^*(f_1+f_2)\right], \tag{1.4}$$

where $B(f_1, f_2)$ is the bi-spectrum in the bi-frequency (f_1, f_2), $X(f)$ is the discrete Fourier transform of time series $x(k)$ and '*' denotes complex conjugate.

The bi-spectrum in equation (1.4) is dependent of the signal energy. The undesired property can be removed by doing the following normalization, defining the bi-coherence function:

$$\text{bic}^2(f_1, f_2) \triangleq \frac{|B(f_1, f_2)|^2}{E\left[|X(f_1)X(f_2)|^2\right] E\left[|X(f_1+f_2)|^2\right]}, \tag{1.5}$$

where “bic” is known as the bi-coherence method. This is bounded between 0 and 1.

Using the bi-coherence function two indices can be calculated, a non-Gaussian index and a non-linear index. For a loop to be regarded as sticky both indices needs to be higher than a given threshold.

Implementing the methods in Matlab can be a bit tricky. For this purpose I recommend the book “Numerical Recipes in Fortran 77” [16]. This was useful for understanding and modifying an implementation already conducted at the lab, for use in chapter 4 for filtering of data.

1.5.3 Relay technique/curve fitting

This method, proposed by Rossi and Scali [20], is based on curve fitting of the recorded signals. Every significant half cycle of the recorded oscillation is fitted by using three different models:

- The output response of a first order plus delay model under relay control.
- A triangular wave.
- A sine wave.

Some error norm is evaluated between the plant data and the fitted model. The model with the smallest norm is said to describe the recorded signal. The relay model and the triangular wave are associated with stiction, whereas the sine wave with the presence of external perturbations.

Also this method have have an index that characterizes the loop as sticky or subject to other phenomena. This index is rather intuitive and will be included here as an example of a stiction index. Let E_S be the minimum square error obtained by the sinusoidal approximation and E_{RT} the one obtained by the better fit of either the relay model or the triangular wave. The index S_I is then

$$S_I \triangleq \frac{\bar{E_S} - \bar{E_{RT}}}{E_S + E_{RT}}, \tag{1.6}$$

where $\bar{x}$ is the mean of x. S_I is normalized to be $S_I \in [-1,1]$, negative values indicate better fit of sinusoids, positive values better fit of the relay model or triangular waves. In the article the uncertainty zone is defined as $|S_I| < 0.21$.

1.5.4 Comparison of the three techniques presented above

Hovd [10] points out that no comparison of stiction detection techniques has been conducted. However, the three techniques outlined above are compared in [20].

A major finding was that all three methods have uncertainty regions where no decision can be taken. In particular for lag dominant processes application of the methods were difficult, sometimes the cross-correlation method gave wrong indication, whereas the other two gave “no indication”. The methods base their analysis on the controller output and the process variable. A possible explanation for why lag-dominant processes were worse than the delay-domant ones is that in especially the relay method the gain of the process variable is important for the curve-fitting. Further one also expects that highly delay-dominant processes could be difficult for analysis, in particular for bi-coherence and cross-correlation techniques, which is based phase-coupling and correlation between the signals respectivly.

For efficiency they found that the bi-coherence method scored best, having a rather short computational time and usually giving the correct judgment. The relay-technique is very time-consuming due to the fitting of the three different models to the data in time domain.

1.6 Methods based on qualitative description formalism

By looking at figure 1.2 on sees that when plotting the valve position versus the controller output a certain pattern is expected to appear. By using our eyes we can easily recognize such patterns, because the human eye is indeed an excellent pattern recognition instrument.[4] In the notion of automatic pattern recognition the object is to enable a computer to solve this task, that is to look at the process variable and control output and then determine if there is a typical stiction pattern in the MV(OP) plot.

The basic idea when doing the pattern recognition automatically is to break the recored signals down into a set of primes. These primers are usually symbols that represent some movement in the data. There are often three steps used to identify patterns in the signals and describe them schematically [18]:

1. Represent both signals (typically noisy) by a set of primers using a chosen scheme. Mathematically speaking, we try to project the recorded signals into the space of the primers.

2. Combing the two symbolically described signals to get a time-development of the MV(OP) plot. If necessary one can from episodes at this point, by collecting movements that are following each other in time. This will later be shown by an example.

[4] Thanks to professor T. Hertzberg at NTNU for telling me this.

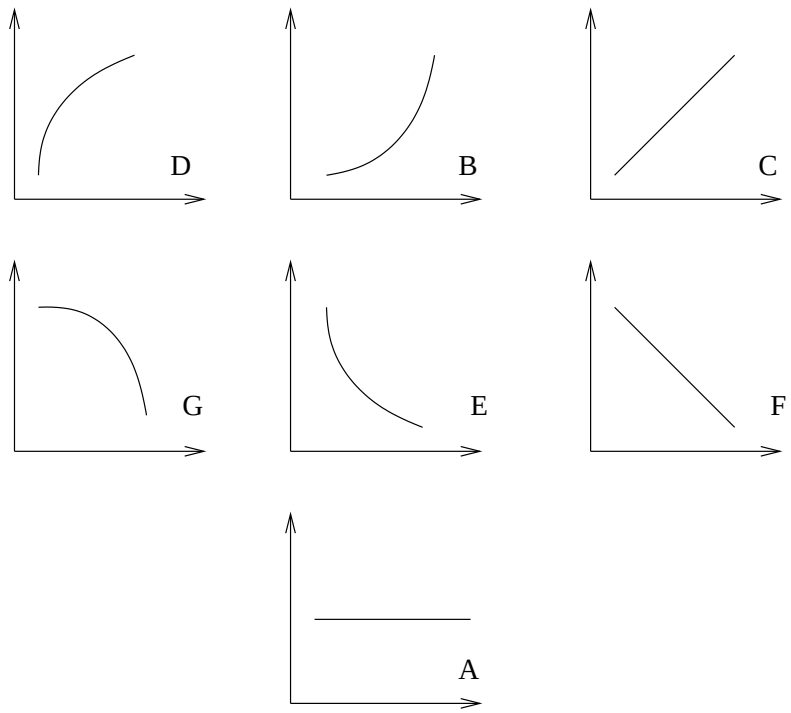

Figure 1.4: Primers used in [18]. Figure redrawn after the same reference.

3. Use a grammar to describe the symbolical time-trend to determine if stiction is present or not.

The most challenging steps are to choose a good set of primers and how to do the projection of the signals into the chosen set. Figure 1.4 shows the 7 primes used in [18]. If these primes defines an orthonormal basis for the space is unknown, but it could be worth investigating. They are included here as an example of such a set of primers.

For identifying the 7 primes it is recommended to use a neural network. The specific network for this case is described in [19]. Using the network requires some programming and tuning. Important points are:

- Fixing the time window for the identification. If the time window is too small, only primer "A" will be identified [18], while a too large time-window will not enable detection of local phenomena.
- Train the network. This procedure is explained in [19]. There is a need to train the network for all the primes using suitable training sets.

Other articles that are more or less based on the same philosophy as described above are [14] and [33]. Reference [32] uses wavelet theory for the identification.

In the latter it is pointed out that a lot of experimentation is needed to used the method described above, and a new design of the neural network is proposed. Introductions to neural networks can be found in [28] and [27].

After reading about the methods above and trying to understand how to implement and use a neural network, the general impression was that these methods require a lot of work to make them functioning. Perhaps this is reflected in the fact that there were few industrial examples in the articles, the methods have been used only on a few selected loops.

1.6.1 Yamashita stiction detection method

A simpler method that possibly avoids the use of neural networks is proposed by Yamashita [34]. In this thesis this method is referred to as the "Yam" method. The concept of this technique is the same as the ones presented above. However, there are differences that made this method more interesting for testing/application on industrial data. These are:

- Only three primitives are used: Increasing (I), decreasing (D), and steady (S).
- There is a method proposed for using the differentials of the recored signals in the symbolic identification. If the signal is very noisy there might be a need to use neural network, but the "default" technique is based on the differentials.
- The "grammar" for interpreting the collected time series is very simple and is described in detail.
- There are very few tuning parameters in the method proposed.

The strongest argument against the method is that it will obviously be sensitive to noise, as the differentials are used as a basis for the symbolic representation. Some industrial examples were given in the original paper, but there was a need to find out how the method performed on real plant data, not just selected loops by Yamashita. This was the main motivation for conducting this work. A review of the basic concepts behind the Yam method will now be given. In chapter 2 results of using the method on simulated and real data are presented.

Assuming now we that have recorded data of OP and MV (or PV for flow loops).[5] The simplest way to describe the trends in the data is to use the three primers increasing (I), decreasing (D) and steady (S). These can be identified by

[5]For flow loops with incompressible fluids this proportionality is good enough. (See also appendix B for information about scaling.)

using the normalized differentials of the recored signals. As proposed by Yamashita, and used throughout this thesis, the standard deviation of the differentials is used as a threshold for the symbolic representation. In words, the identification works like this:

1. Calculate the differentials of the given signals.
2. Normalize the differentials with the mean and standard deviation. (By subtracting the mean and dividing by the standard deviation.)
3. Quantize each variable in three symbols using the following scheme (x is the recored signal and $\dot{x}$ is the normalized differentials):
   - If $\dot{x} > 1$, x is increasing (I).
   - If $\dot{x} < -1$, x is decreasing (D).
   - If $-1 \leq \dot{x} \leq 1$, x is steady (S).

It could be worth noting that the normalization is invariant to scaling of the variables. This implies that normalizing the signals by their range does not influence the method. This is proved in appendix B.

Figure 1.5 shows and example on how the method ideally should work on industrial data for a sticky valve. As indicated in the figure, when the valve is jumping up or down, we want $\left|\frac{\partial \mathrm{MV}^*}{\partial t}\right| > 1$, while $\left|\frac{\partial \mathrm{MV}^*}{\partial t}\right| \leq 1$ when the valve is steady. The symbol * means normalized by subtracting the mean and dividing by the standard deviation. The idea is that when the valve is jumping the MV-signal changes more than we the data is just changing due to noise, so for this to work the jumps needs to "stand out" form the noise. In this case the data was rather noisy, and most likely some of the time when the valve was stuck (seen by visual inspection) it was wrongly indicated as moving. As for OP, we want it to be characterized as either increasing (I) or decreasing (D) when the control output is integrating and the valve is stuck.

Assuming now that the identification works good, the next step is to combine the symbols for OP and MV to get a symbolic representation of the development in an (OP, MV) plot. The primes for the combined plot are shown in table 1.1, and displayed graphically in figure 1.6. The primers are of the type τ_{xy}, where x is a movement in the OP direction and y is a movement in the MV direction. The "sticky" primers are the ones when the controller is changing its output while the valve position is unchanged. These are framed in the table, and are $\boxed{\mathrm{IS}}$, and $\boxed{\mathrm{DS}}$.

Based on this, a stiction index ρ_1 can be defined:

$$\rho_1 = (\tau_{\mathrm{IS}} + \tau_{\mathrm{DS}}) / (\tau_{\mathrm{total}} - \tau_{\mathrm{SS}}). \tag{1.7}$$

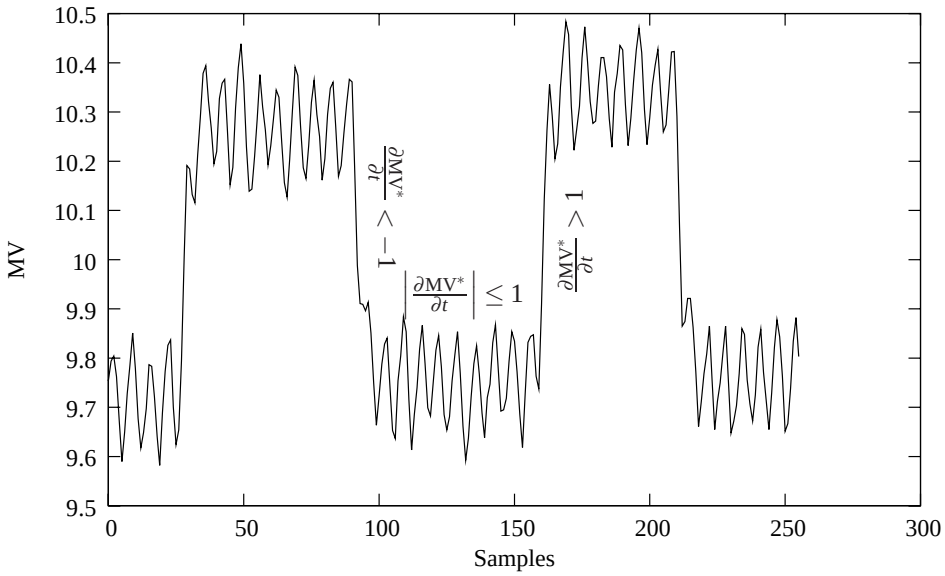

Figure 1.5: Idealized symbolic representation by using the standard deviation of the normalized differentials as threshold. This data was regared as rather noisy.

Table 1.1: Primers for an (OP, MV) plot. See also figure 1.6.

OP/MV	D	S	I
I	ID	IS	II
S	SD	SS	SI
D	DD	DS	DI

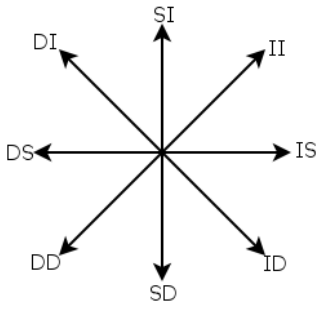

Figure 1.6: Qualitative movements in $x-y$ plots.

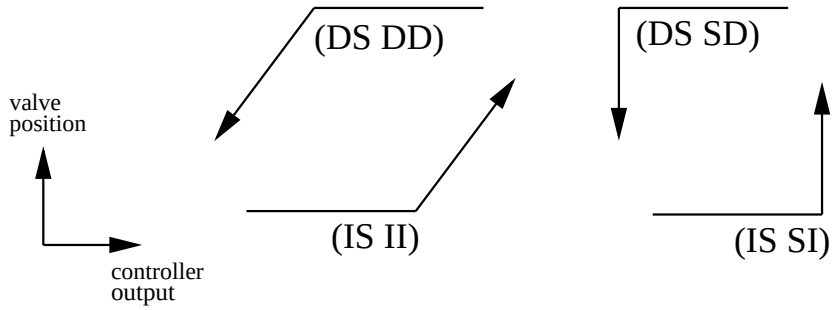

Figure 1.7: Typical patterns in loops affected by stiction described in the combined plot. Adopted from [34].

This equation is the "grammar" of this method, since is interprets the combined symbolic representation of the signals into an index that should tell if stiction is present or not. In (1.7), τ_{IS} is the total number of occurrences of the combined primer "IS", and so on. Note that the time when both signals are steady is removed. After removing τ_{SS}, the sticky movements framed in table 1.1 corresponds to 2 of 8 symbols. With only random signals $\rho_1 \approx 0.25$, so a loop with $\rho_1 > 0.25$ can be regarded to be sticky.

In applications the ρ_1 index is found to be to coarse to be useful. This will later be shown in examples with application to industrial data. Often the index is high even though there is no evident stiction pattern in the plot. This calls for a refined index. Figure 1.7 shows the typical patterns expected in a sticky valve. (See also chapter 3 for a discussion about the different expected patterns.) Based on the patterns shown in figure 1.7, the following refined index ρ_2 can be defined:

$$\rho_2 = (\tau_{\text{IS II}} + \tau_{\text{IS SI}} + \tau_{\text{DS DD}} + \tau_{\text{DS DD}}) / (\tau_{\text{total}} - \tau_{\text{SS}}). \tag{1.8}$$

In (1.8), $\tau_{\text{IS II}}$ is the total number for IS samples in all the found (IS II) movements in the observation window, $\tau_{\text{DS DD}}$ is the total number for DS samples in the found (DS DD) movements, and so on. For example, if we had the time series (IS IS IS IS II II II) $\rightarrow (5 \cdot \text{IS}, 3 \cdot \text{II})$[6], $\tau_{\text{IS II}} = 5$. Further $\rho_2 \leq \rho_1$, where the equality holds when all the sticky motions corresponds to the shapes shown in figure 1.7.

In the extreme case when the valve does not move, only patterns IS, SS and DS will be found. This special case will make $\rho_2 = 0$, not 1. To avoid this a new index ρ_3 is used, that is calculated by subtracting all the sticky patterns that does no match the patterns shown in figure 1.7 from ρ_1:

$$\rho_3 = \rho_1 - \left(\sum_{x \in \mathscr{W}} \tau_x \right) / (\tau_{\text{total}} - \tau_{\text{SS}}). \tag{1.9}$$

[6]This is what is know as "episodes".

The set $\mathscr{W}$ contains all the sticky patterns that have noting to do with stiction. In symbols, these are $\mathscr{W} = \{$IS DD, IS DI, IS SD, IS DS, DS DI, DS SI, DS ID, DS II, DS IS$\}$. For example, if the movement was (IS IS IS DD IS IS II) $\rightarrow (3 \cdot \text{IS}, 1 \cdot \text{DD}, 2 \cdot \text{IS}, 1 \cdot \text{II})$, we should subtract $3/7$ from the original $\rho_1 = 5/7$, because the three first IS primes could no be a part of a stiction pattern since they were followed by a DD.

To summarize, the method its schematic implementation is like this [34]:

1. Obtain a time series of the controller output and valve position (or corresponding flowrate).
2. Calculate the time difference for each measurement variable.
3. Normalize the difference values using the mean and standard deviation.
4. Quantize each variable in three symbols.
5. Describe qualitative movements in (x,y) plots by combining symbolic values of each variable.
6. Skip SS patterns for the symbolic sequence.
7. Evaluate the index ρ_1 by counting IS and DS periods in the patterns found.
8. Find specific patterns and count stuck periods. Then evaluate the index ρ_3.

The method can easily be implemented in any suitable programming language. In appendix A an implementation in Matlab is shown.

Testing and evaluation of the method will be discussed in chapter 2.

1.7 Control loop performance monitoring

This section will try to show how stiction detection could be a part of a larger contol loop monioring system. A general introduction to the monitoring problem will also be given.

Hovd [10] says that there are many reasons why decreased performance is allowed to last in a plant. Even though the operator might detect the degradation in performance he or she might not report this to the automation department. Often the operators learn to cope with the plant as it is. Further they lack the competence to determine what is good and bad performance. When asked a general question about whether the control performance it acceptable, they may confirm that the control is good, even though this is not the case.

The control department on the other hand usually has too few resources to investigate all the loops at a high frequency. As this department usually is very small, they must use their time to keep the various automation and control systems in operation [10]. Further the employees are often inexperienced to assess the performance of the control system, as running the control system is their main focus. After an initial commissioning phase most controllers are usually "left alone" for a long period of time [10].

Listing all the reasons for justifying the development of a control performance monitoring (CPM) system is a time-consuming task, and I will stop with the items listed above. More justification can be found for example in [4, 7, 20].

A good CMP tool should help the control engineer to [10]:

- focus on where the control problems are most acute
- diagnose the cause for poor control performance
- quickly assess whether significant improvements are easily achievable, e.g. by retuning the controller or applying methods to reduce the effects of stiction.

The group here in Pisa has developed the PCU tool which may be regarded as a CMP tool as defined above. The system is described in detail in [23], though still work is in progress and the properties of the system are changing. A recall of the important aspects enlightened in [23] will now be conducted.

1.7.1 PCU

The architecture of the PCU tool is shown in figure 1.8. It consists of three modules. These are:

Module 1 Determine if the response is oscillating by using a modified Hägglund index, or if the response is sluggish. If neither of the two, the controller is performing good.

Module 2 For the case of an oscillating response, it will be considered to be caused by poor controller tuning if the response is damped. Also sluggish responses are said to be due to poor controller tuning. In this case an identification and retuning program is stared.

Module 3 For the case of an oscillating, non-damped response, there are five possible outcomes for the analysis.

- No dominant frequency in the spectrum of the signals: The oscillation is caused by an irregular disturbance (i).

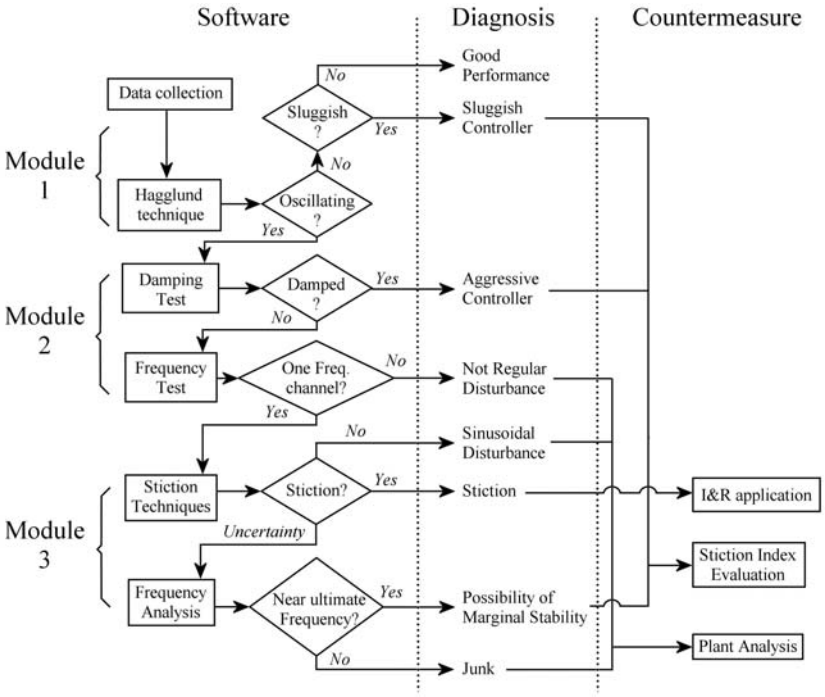

Figure 1.8: The architecture of the PCU. Adopted from [23].

- Dominant frequency found: Three different stiction detection algorithms are applied to the data. These tests are different in their need of computational time and are applied in series. (They use the three "popular methods" described earlier in this chapter.) If stiction is detected in at least two of the tests the loop is said to be sticky (ii). The three stiction detection methods implemented can also characterize the loop as being subject to a disturbance (iii). If this is the case, the loop is tested for marginal stability. If the outcome of the stiction detection is "uncertain", also marginal stability is checked. The possible outcomes are that the loop is either subject to a regular disturbance (iv), the loop is marginally stable or non conclusion is found (the loop is assessed as uncertain) (v).

With this architecture it is possible to both detect abnormalities and determine the cause. The identification and retuning module should propose a new tuning of the controller. If the loop is considered to be sticky there is an index calculated in order to determine how much time will go before the performance of the loop become unacceptably bad [21].

The software is still under development. For the time of writing this thesis we were focusing of the stiction part of the package. As properly noted in [20], the stiction detection techniques have regions of insecurity and may sometimes give the wrong decision. This motivates the need for more good methods to be included in the package. Since the Yam method is recently published is it therefore interesting to see if this method can be included.

Neither of the current stiction detection algorithms have a robust way of assessing the amount of stiction, once found. A method for this is proposed in [4], and will be investigated in chapter 4. Today the only implemented method is the one published in [21]. However, as with the stiction detection techniques, it would be advantageous to have more routines estimating the quantification.

Once stiction has been identified and a quantification is conducted it can be interesting to suggest some measures to take to run the valve with a performance good enough to avoid unnecessary plant shutdowns. Two possible techniques for this is the knocker-technique proposed by Hägglund [8], and the forcing of high-frequency inputs in [2]. A more comprehensive method can be found in [15], which proposes a method with re-design of the controller. Of course, when the dynamics of the stiction is determined one can design a non-linear "feed-forward" controller to overcome the stiction. Due to its high complexity, I doubt that this is wanted in the chemical industry. If time allowed some work on this subject should be included in this thesis. Unfortunately only the literature search was conducted, so this is still an open point.

Chapter 2

Yamashita stiction detection method

Most of this work is already published in the ANIPLA conference paper, which is attached in appendix E. This chapter will be a slightly longer version of that paper. The introduction part of the paper is already rewritten in a rather longer form in chapter 1. In that chapter the basic concepts of the method were reviewed, and readers are encouraged to use that chapter as a reference for the present discussion.

The method can easily be implemented in Matlab. A possible implementation is shown in appendix A. Due to demand by users a simple GUI was developed. A small discussion about how this is working and its implementation as well as some screen-shots are included in appendix A.2. In the GUI code it is also shown how data easily can be imported from Excel files, something that was used throughout this work as the plant data were stored in Excel files. Finally, an example of how the method can be implemented in PCU was made. This is shown in appendix A.3.

This chapter contains the following sections: First, application of the method to simulated data. Then follows a application to plant data and last some general conclusions about the technique.

2.1 Application to simulated data

The main motivation for testing the technique on simulated data is to understand how it performs in various cases. For a possible industrial implementation it is important to understand how noise, external disturbances, and set point changes affects the performance and efficiency of the method. Therefore, in this section “extreme” conditions of what is expected from plant data will be investigated. Afterward the method will be applied to plant data.

As previously noted, the model developed by Choudhury is used to generate simulated data for which the method it tested. Figure 2.1 shows this model

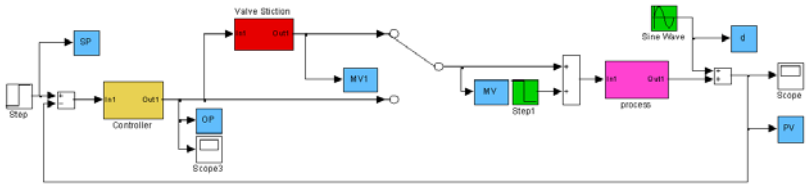

Figure 2.1: Snap-shot of simple Choudhury model implemented in Simulink.

implemented in Simulink. A shown in figure 1.2, the size of the deadband plus stickband S and the size of the slip-jump J needs to be specified. The process model used was a first order plus delay model (FOPD), $G = ke^{-\theta s}/(\tau s + 1)$. For the controller a SIMC-tuned controller was used [25]. For a FOPD model the corresponding PI-controller tuning with control equation $c(s) = K_c\left(\frac{\tau_I s+1}{\tau_I s}\right)$, the tuning rules are:

$$K_c = \frac{1}{k}\frac{\tau}{\tau_c + \theta} \tag{2.1}$$

$$\tau_I = \min\{\tau, 4(\tau_c + \theta)\}. \tag{2.2}$$

The only tuning parameter is τ_c, and this will be set equal to the delay $\tau_c = \theta$, which is the recommended setting [25].

Before applying the Yam method on simulated and plant data, at least three degrees of freedom needs that to be specified:

- Length of time window
- Threshold in symbolic representation
- Sampling time

Except for practical problems there is no upper limit for the time window. In this section there were always at least 3 cycles of oscillations. For the threshold the standard deviation of the time differentials was used, as recommended by Yamashita. The effects of altering the sampling time will be investigated in this section by altering the sampling time in the presence of noise in the recorded data.

Yamashita [34] writes that some prefilter can be used if the raw signals are very noisy. This was not used here, as keeping the method as clear and simple as possible was one of the motivations for choosing the Yam method.

In the simulated data MV is available, and in all of this section this data-set will be used instead of PV as a basis for calculating the indices ρ_1 and ρ_3.

2.1.1 Noise-free data

In [20] a comparison between the three "popular" methods described earlier (cross-correlation, bi-coherence and relay) is conducted. We wanted to test the Yam method on the same conditions as in that paper (reference to figure 5 in the paper). They investigated the window ($1 \leq S/(2J) \leq 7$, $0.1 \leq \theta/\tau \leq 3$). In this window there are actually 4 degrees of freedom, because the variables does not appear as these ratios in the equations, indeed the model describing stiction is rather non-linear in all four variables. Further it was not specified how they tuned the controller from time to time, which introduces two more degrees of freedom for a PI controller. A last degree of freedom is the process gain, k, but it is assumed that this was set to 1 in their article. To avoid further confusion, in this section the parameters are $\{k, \tau, J\} = \{1, 10, 1\}$ for all processes, and the SIMC-tuned PI controller is used.

With the aid of some "for"-loops the method was examined on the window defined above. The result was very encouraging, as all the cases resulted in a $\rho_3 > 0.9 \gg 0.25$. The calculated ρ_3 indices are shown in figure 2.2.

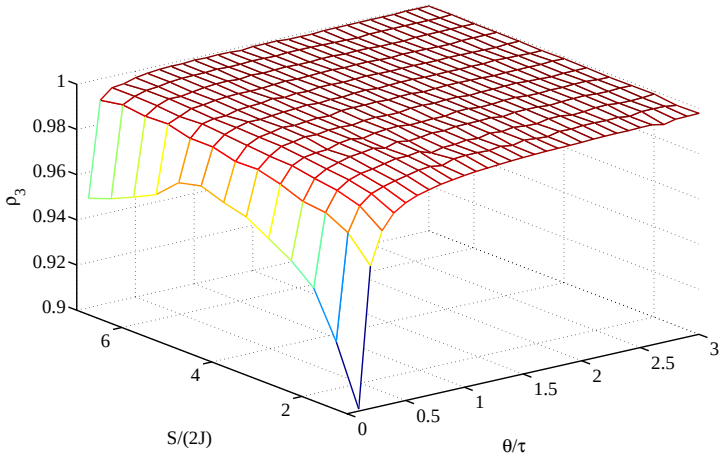

Figure 2.2: Evaluation of stiction index ρ_3 on the FOPD process $g(s) = \frac{k}{\tau s - 1} e^{-\theta s}$ with $\{k, \tau\} = \{1, 10\}$ for all processes. In the stiction model $J = 1$.

This implies that in practice no region of uncertainty exists, as this requires that $\rho_3 \to 0.25$. Hence the Yam method seems a lot more reliable than the methods compared in [20] However, this comparison is already biased toward the Yam method. In the simulation MV was available and this was used together with OP as a basis for calculation of the indices ρ_1 and ρ_3. The methods compared in [20] had PV and OP data available, which really is another situation. It is well-known that the process variable PV is the valve-position shifted in phase and gain, hence

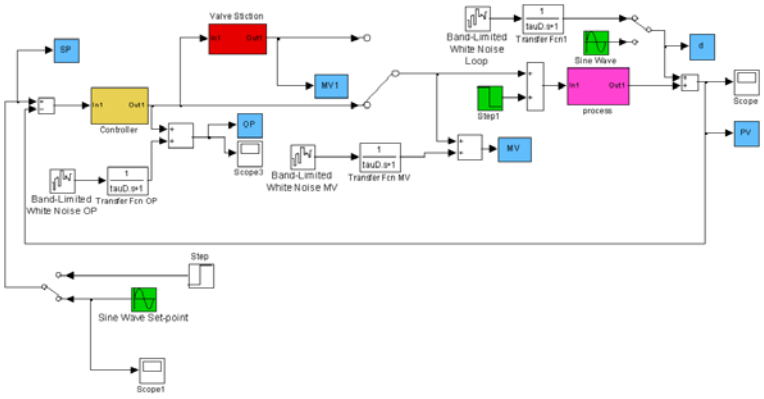

Figure 2.3: Choudhury model with noise-blocks added at the measurements and at the reference.

the information-context is distorted when using PV instead of MV. Though the comparison is biased this result was so encouraging that further investigation of the technique was desired.

In this section the sampling time was set to 0.2 time-units, which may be too low for practical purposes. The next section includes a discussion on the effects of altering the sampling time.

2.1.2 Adding noise

Noise was added to the measurements of MV and OP to investigate the performance with noise present. A picture of the modified Simulink model is shown in figure 2.3. First we tried to add noise directly through the loop, using the "Band-limited white noise Loop" block as seen in the upper right corner of the figure. However, this caused significant convergence problems for the model. This is expected, as Choudhury [5] writes that:

> In order to handle stochastic inputs, the model requires the implementation of a PI(D) controller including its filter under a full industrial specification environment. Alternatively, an exponentially weighted moving average (EWMA) filter placed right in front of the stiction model can be used to reduce the effect of noise.

Neither of the two approaches seems so simple as just adding noise directly to the measurements, so this method was in the end chosen. Only PI controllers were studied in this section, so keeping the loop noise-free and rather adding it to the measurements should not make the results deviate much from the other possible

approach with sending the noise around the loop. If derivative action was present in the controller perhaps the placement of the noise-blocks could have some effect.

Now a small discussion on the chosen way of simulate the noise will follow, thereafter to the results of the simulation.

Band-limited white noise In Simulink, one can simulate the effect of white noise by using a random sequence with a correlation time much smaller than the shortest time constant of the system. The documentation says that for accurate simulations, use a correlation time much smaller than the fastest dynamics of the system. One can get good results by specifying

$$t_c \approx \frac{1}{100}\frac{2\pi}{f_{\max}},$$

where t_c is the sample-time setting for the Simulink block and $f_{\max}$ is the bandwidth of the system in rad/seconds. Since we have chosen to use a SIMC-tuned controller in our simulations, we know that the controller bandwidth is approximately $w_B \approx 0.5/\theta^{\text{effective}}$ [26]. The effective delay $\theta^{\text{effective}}$ may be found by using the half-rule [26], but since we are using a FOPD model the effective delay is simply the delay in the process, $\theta^{\text{effective}} = \theta$. We will use this bandwidth as the assumed system bandwidth in the specification of t_c. The **noise power** is the height if the power spectral density of the noise. The covariance of the block output is the **noise power** divided by t_c.

The noise was filtered with a first order filter $1/(\tau_F s+1)$. This block is shown in figure 2.3, being between the band-limited with noise and the measurements. The noise power was tuned to be about equal for the OP and MV measurements. This resulted in a noise to signal ratio of about 0.1 to 0.2 for both signals. Then simulations were conducted altering the filter time constant τ_F and the sample time T_s, keeping the noise power block unchanged. Results of running the Yam method on the simulated data are shown in table 2.1. To investigate the robustness of the method the indices were also calculated for loops without the presence of stiction.

Figure 2.4 shows an example of the process studied in table 2.1. Here the parameters were $\{S,J,k,\tau,\theta\} = \{6,1,1,10,10\}$. The sample time was 10, which is equal to the time constant of the process. This is a recommended value for sampling in process analyzers, for parameter estimation it is optimal to set the sampling time equal to the dominant time constant of the system [11]. In this case also the noise filter constant τ_D was set to 10 ($= \tau$). This means in that the high-frequency part of the noise should be filtered out.

In the figure both OP and MV and their normalized derivatives are shown. The stiction pattern (right part of the figure) is very clear in this case, both with and

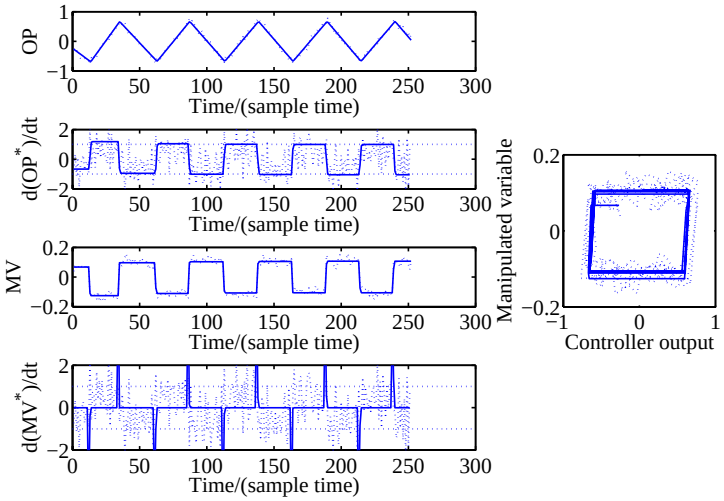

Figure 2.4: Simulation with and without noise. The dotted lines are the noisy signals. OP, MV and their normalized derivatives are shown. The horizontal lines are the threshold ($= 1$) for symbolic representation.

without noise present. To get an understanding for how the Yam method works and why it is inherently sensitive to noise it is helpful to look closely at first the signals and then their normalized derivatives. The signals with noise are almost equal to the signals without noise. However, taking the derivative one sees that the noise is more evident and the noise-influence is (as expected) enhanced in this case. (Remember that we want $\left|\frac{\mathrm{dMV}^*}{\mathrm{d}t}\right| > 1$ only when the valve is jumping. Further, in this example, we always want $\left|\frac{\mathrm{dOP}^*}{\mathrm{d}t}\right| > 1$ for the indentification to work good.)

Now, a more global view. By considering the simulation results for the process/stiction conditions shown in table 2.1 the following observations can be made:

- The frequency-content of the noise is significant. Adding much high-frequency noise makes the method unable to detect stiction. This is evident by looking at the columns in table 2.1 with $\tau_F/\tau = 0.1$ and $\tau_F/\tau = 0.01$. The results for simulations with and without stiction are about equal.

- The sampling time is important. Lowering the sampling time makes the method inefficient, as the calculation of the differentials will be too dominated by the noise. Setting the sampling time very high is also disadvantageous, so there must be an optimum where we avoid sampling too much

Table 2.1: Results altering the filter constant τ_F and the sampling time T_s. The upper table shows simulations with a sticky valve, whereas the lower table shows simulations without stiction. The numbers are the values of ρ_3 (ρ_1).

		τ_F/τ – with stiction		
		0.01	0.1	1
$\frac{T_s}{\tau}$	0.01	0.12(0.40)	0.18(0.40)	0.30(0.62)
	0.1	0.10(0.40)	0.14(0.42)	0.30(0.65)
	0.5	0.08(0.41)	0.12(0.42)	0.42(0.73)
	1.0	0.11(0.46)	0.14(0.45)	0.59(0.79)
	10	0.27(0.6)	0.20(0.40)	0.50(0.50)

		τ_F/τ – without stiction		
		0.01	0.1	1
$\frac{T_s}{\tau}$	0.01	0.13(0.40)	0.18(0.40)	0.19(0.40)
	0.1	0.10(0.40)	0.13(0.42)	0.17(0.40)
	0.5	0.08(0.40)	0.12(0.41)	0.13(0.35)
	1.0	0.11(0.44)	0.11(0.43)	0.17(0.41)
	10	0.07(0.53)	0.18(0.41)	0.18(0.53)

noise but still observe the stiction induced limit cycle. From these data, setting the sample time equal to the dominant time constant seems to be a good default setting. Theoretically the sampling time should be in the frequency domain of the limit cycle and not of the noise.

- For the cases studied with no stiction in the loop, ρ_1 is always too high, whereas ρ_3 correctly rejects stiction in all cases. This implies that we should use ρ_3 as the determining index, not ρ_1.

From these observations one can conclude that the method is sensitive to noise, and there exists an optimal sampling time. It seems like a good default setting for this sampling time is to set it equal to the dominant time constant of the process, which agrees nicely with [11].

2.1.3 Varying set points

An inner PID controller in a conventional cascade is an example of a controller where the set point can be subject to frequent changes. A loop with severe stiction and subject to rapid set point changes found in plant data is shown in figure 2.5. A simulated example is shown in figure 2.7. Here the set point was changing linearly with the function $y_{sp}(t) = 0.5\sin(0.001t)$. This frequency should be well inside the bandwidth of the controller. With a $T_s = \tau$ we found stiction indices

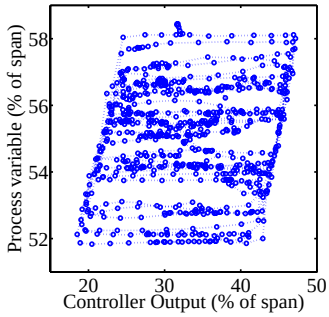

Figure 2.5: Loop affected by both stiction and set point changes. From plant data. 720 samples are plotted.

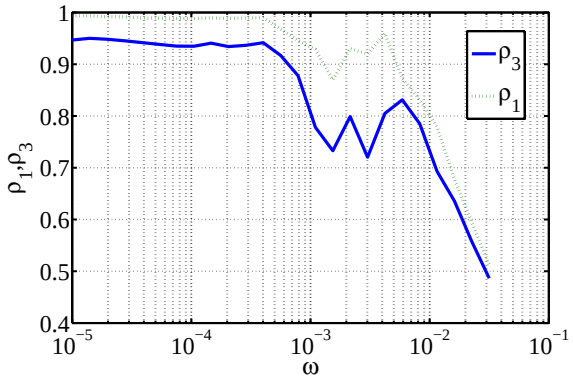

Figure 2.6: Indices ρ_1 and ρ_3 as a function of set point-change frequency, $y_{sp} = 0.5\sin(\omega t)$. There was no noise added and $T_s = 0.1\tau$.

$\{\rho_1, \rho_3\} = \{0.71, 0.70\}$, which clearly indicates stiction. On this noise-free signal we get better results by reducing the sample time, for instance with $T_s = 0.1 \cdot \tau = 1$ we get $\{\rho_1, \rho_3\} = \{0.94, 0.93\}$, but most likely this sample-time is more sensitive to noise.

Another typical example is a PID controller receiving commands from an advanced process control system (APC). A reasonable time-scale separation between the control layers in a hierarchical structure should be about 5 or more in terms of closed loop response time [26]. When the frequency of set point changes increases from low frequencies to higher, the indices are expected to decrease, because the controller needs to work more to follow the command signal, and hence the stiction pattern will be less clear.

Figure 2.6 shows the calculated indices while varying the frequency of set

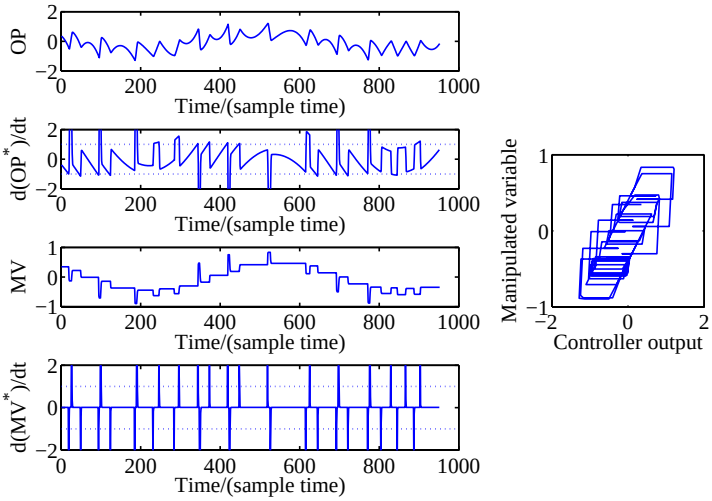

Figure 2.7: Simulation of set point changes. $y_{sp}(t) = 0.5\sin(0.001t)$. The plots show OP and MV signals and corresponding normalized derivatives used in calculation of the symbolic representation. The process parameters were $\{S, J, k, \tau, \theta\} = \{6, 1, 1, 10, 10\}$.

point change for a loop with parameters $\{S, J, k, \tau, \theta\} = \{6, 1, 1, 10, 10\}$. A SIMC-tuned controller with an assumed closed-loop bandwidth of $\omega_B \approx 0.5/\theta$ [26] was used. For a well designed cascade we expect set point changes at a frequency lower than $(1/10)(1/\theta)$, which will be the assumed bandwidth of the outer controller. (Proof: Let the inner and outer loops have expected closed loop response times τ_{c1} and τ_{c2} respectively. Assume that both of them are SIMC-tuned controllers. We then have that the assumed bandwidth for the outer controller is $\omega_B^{\text{outer}} = \frac{1}{2}\frac{1}{\theta_{\text{eff}}^2} = \frac{1}{2}\frac{1}{\tau_{c2}} = \frac{1}{2}\frac{1}{5\tau_{c1}} = \frac{1}{2}\frac{1}{\theta_{\text{eff}}^1} = \frac{1}{10}\frac{1}{\theta}$. θ_{eff}^1 and θ_{eff}^2 are the effective delays in the inner and outer loops.) So, in the case study we expect set point-changes with maximum frequency of about $(1/10)(1/\theta) = (1/10)(1/10) = 0.01$ radians/second. By looking at the fig. 2.6 one observes that the indices are relatively high up to this expected frequency, where a decrease in the indices follow for higher frequencies. The stiction parameters were the same in all the cases, but the presence of high-frequency set point changes makes the indices decrease.

This analysis shows that for well-tuned cascades, linearly changing set points within the bandwidth should not be able to deteriorate the performance of the Yam method significantly. If the loop is affected by higher frequency set point changes than what it was designed for (or sustained changes around the bandwidth frequency) the Yam method may not be able to detect a possible presence of stiction.

An advanced process control system (APC) can give commands to the layer below in a step-wise fashion. For fast loops, such as flow loops, these steps may propagate to steps in the OP and MV. A simulation was conducted to investigate this effect. If was evident that introducing steps in the input made the steps dominate the differentials of the recored signals. In words, the signals were only increasing (I) or decreasing (D) when the steps occurred, otherwise always steady (S). To understand the significance of this observation to industrial usage, investigation of plant data are necessary. The sharp steps introduce problems, but if they occur rarely or filtered it may still be possible to use the method. (For example by "bump-less transfer".)

It should further be noted that using the Yam method on "bad data", where the set point is changing to rapidly, is not really dangerous because the ρ_3 index is low in these cases. This simply means that no indication is given, but this is a lot better than if the method would indicate stiction for the bad data.

Yet another issue worthwhile noting is that the Choudhury model can be faulty in the case of high-frequency disturbances/set-point changes without "a full industrial specification implementation" of the PID controller. See more details at page 26. This can explain the non-smoothness of the plot in figure 2.6, page 30.

An alternative physical explanation for the observed decrease in ρ_3 in figure 2.6

When set point changes around the bandwidth are introduced, these changes will of course keep the valve moving for longer periods of time. For the valve to stick its velocity needs to be lower than some given threshold. Keeping it moving avoids it to stick. This agrees with the proposed method of introducing fast, forced cycles at the input with a higher frequency than those generated "naturally", in reference [2]. If this method is applicable to real valves is an important still unanswered question. In [8] a high-frequency zero-mean signal is added to a pneumatic valve (called "dithering the valve"). Quoting the reference, "dithering is unfortunately not useful to overcome stiction in pneumatic control valves. A dithering signal may perhaps be generated by the positioner, sine the pilot valve is rather fast. However, this high-frequency signal will be low-pass filtered (integrated) in the actuator. Since the output from the pilot valve furthermore is limited in amplitude, it is not possible to generate a dithering high-frequency pressure drop over the actuator piston".

2.1.4 Conclusion after using the method on simulated data

For the noise-free case the technique performs well. As noise is introduced, performance decreases. The sampling time may affect the performance of the

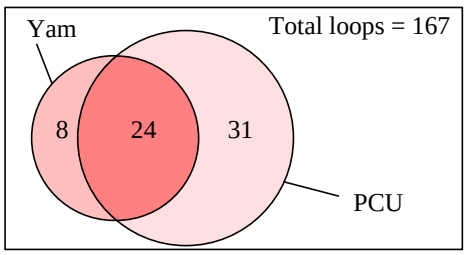

Figure 2.8: Loops found to be sticky by the Yam method and the PCU.

Table 2.2: PCU report for the 8 loops where Yam reported stiction (see figure 2.8).

Verdict by PCU	Number of loops
Good performance	1
No dominant frequency	7

method. Setting the sampling time too low (or too high) introduces problems for stiction detection. For cascaded loops, simulations shows that as long there is a time-scale separation of 5 or more between the layers, the method should still be able to detection the presence of stiction in the inner loop.

The real test of the method will be plant data, then one will know if the noise level is too high for application of the method or not.

2.2 Application to plant data

A total set of 216 industrial PID loops, of which 167 were flow loops, were available for analysis. The sources were different italian chemical plants, the data have been gathered at the lab for some time before and during my stay here.

All the loops analyzed were compared with a program called Plant Check-Up (PCU), a prototype for stiction detection in industrial use. The architecture of PCU is shown in [22]. For stiction detection it uses the cross-correlation method [9], the bi-coherence method [4] and the relay technique [20].

2.2.1 Results

In the industrial data set of 167 flow loops the Yam method reported stiction in 32 of the loops, while running the PCU on the same data resulted in 55 loops reported as sticky. This is illustrated in figure 2.8, where one also observes that 8 loops were found to be sticky by the Yam method but not by the PCU.

Table 2.2 shows the report from PCU these 8 loops. When the PCU reports

"No dominant frequency" it can not find a dominant frequency of the signals and it does not initiate the stiction detection module. The relay technique requires this frequency. By running only the bi-coherence method, which does not require a dominant frequency, all of these 7 loops were reported to be sticky, therefore these loops can be considered sticky.

For the loop reported to be performing good, data for more weeks were available. For other weeks, the loop was reported to be under presence of stiction by the PCU, therefore this loop is a limit case.

Of the remaining $168 - 33 = 135$ loops for which the Yam method did not report stiction, 31 were found to be sticky by the PCU. If the PCU is regarded as being correct, one can say that the Yam method detects stiction in about half of the cases where PCU detects stiction. The PCU is more advanced, as it has 3 methods implemented for stiction analysis, so it is expected that not all cases can be detected by the simple Yam method.

A visual inspection of the data can be performed on a computer by displaying the recored data in a (OP, MV) plot that evolves with time. Using this tool, it was evident that for the cases where the Yam method reported stiction the expected pattern as shown in figure 1.2 was shown. (The tool is implemented in the GUI, see appendix A.2.)

Several phenomena were observed for the 31 loops where only PCU found stiction (see figure 2.8). For some of the loops the signals were distorted by noise and no clear patterns were observed. For other loops clear patterns could be observed, but their properties were not of the typical stiction pattern type (see figure 1.2). Often the patterns were similar to an ellipse with no clear parts where the OP was increasing or decreasing with steady MV. A closer look at the patterns observed is given in chapter 3.

2.2.2 Sampling time

Application of the method on simulated data showed that there were both lower and upper limits on sample time. The lower is due to sensitivity to noise. For all the loops the sample time was originally 10 seconds. Let x be a vector of observations with sample time T_s. A naive way to simulate a sample time of $2 \cdot T_s$ is to use every second point in x as a basis for calculation of the indices.

Table 2.3 shows the result of increasing the sample time by a factor 3 and 6. One observes that the method is a bit sensitive to alterations of the sample time, considering that the number of loops detected changes. However, by inspection of the loops reported to be sticky after altering the sample time, both visually and with the PCU, the conclusion was the same for the sample times $T_s/T_s^{\text{original}} \in \{1,3,6\}$. When the Yamashita method reports stiction, a visual inspection of the

Table 2.3: Effects of altering the sampling time.

$T_s/T_s^{\text{original}}$	Loops with $\rho_3 > 0.25$
1	32
3	38
6	34

(OP, MV) plot shows clear signs of stiction. The loops that were changed from not being sticky to sticky by increasing the sample time, due to their increase in ρ_3 to above 0.25 from below, had the same stiction-pattern properties as the original 32 loops.

Theoretically, at least two properties of the method are affected by the sample time, giving upper and lower bounds.

- Given a signal with some noise, reducing the sampling time too much will cause the calculation of the derivate to be completely dominated by the noise.[1]

- Given a more reasonable sampling time, the noise may act as making the method more robust. This is because the presence of noise will increase the standard deviation, hence increasing the threshold for identification of sticky valves. This is desirable, because the method will tend to select the valves with most severe stiction. Setting the sampling time too high remove this feature, but perhaps more importantly with a very large sample time (larger than the frequency of the limit cycle) will disable the method to detect stiction. (With a too high sample time the jumps in the valve will not be detected.)

2.2.3 Minimum observation window

It is interesting to quantify the sufficient length of the observation window, as this is an important practical factor when using the method in applications. The typical length of observation for the plant data was about 6000 samples with an interval of 10 seconds, corresponding to an observation window of 17 hours.

To simulate the effect of having less data we simply removed the ends of the data sets, lowering the available amount to data for analysis by the Yam method.

[1] To understand why, think of a signal with a high-frequency and a low-frequency movement. If we are sampling with a high frequency we will see how the high-frequency part of the signal changes from sample to sample. Sampling with a frequency corresponding to the low-frequency part will reveal how the signal changes with the low-frequency, but we will not observe the high-frequency part.

Table 2.4: Loops reported to be sticky while changing the length of the observation window

Maximum number of samples	$\rho_3 > 0.25$
total length available	32
2000	30
1000	28
500	29
100	30

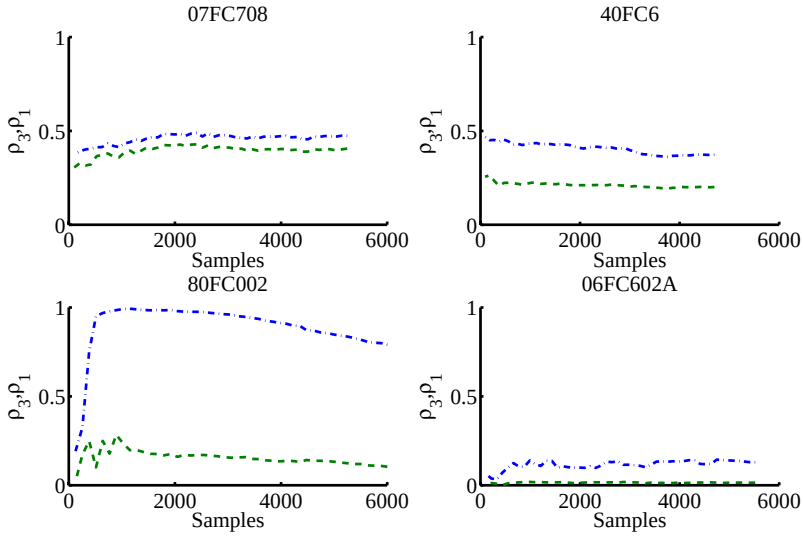

Figure 2.9: Indices ρ_1 and ρ_3 for 4 loops as a function of available samples for calculation of the indices. The figures should be read from right to left to see the effect of reducing the number of samples. The upper curve is always ρ_3.

From table 2.4 one sees that there is not much differences in the results by lowering the allowed samples from unrestricted down to 500 or 100. With a sample time of 10 seconds we could consider sampling 720 samples, corresponding to an observation window of 2 hours.

The effect of reducing the number of samples available to analysis is shown graphically in figure 2.9 for 4 loops. Loop 07FC708 is a loop with a typical stiction pattern, though with a rather noisy flow-signal. Loops 40FC6 and 80FC002 were classified as not being sticky in the initial analysis, their ρ_3-values were both 0.21. 80FC002 may be regarded as an especially "difficult" loop from a Yamashita point of view, due to the large reduction from $\rho_1 \rightarrow \rho_3$. In fact, this was the loop with the largest reduction of all the loops inspected. Loop 06FC602A had a initial

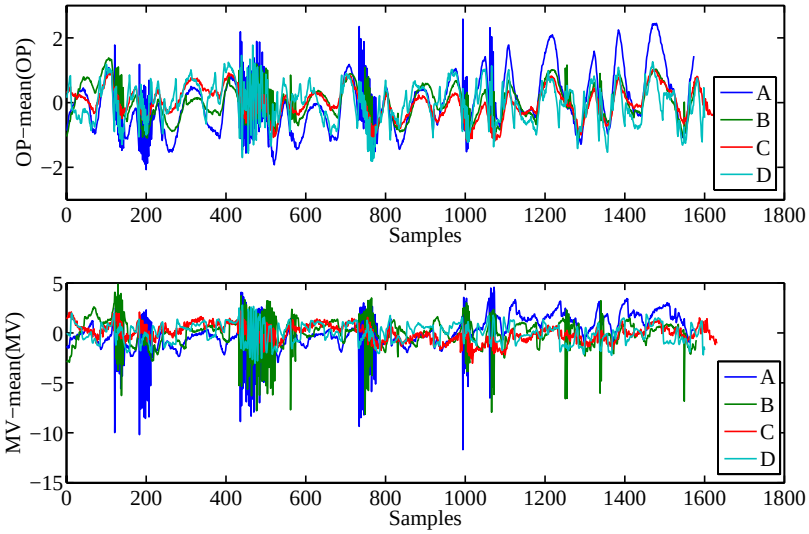

Figure 2.10: Plots of OP and MV (which is the PV) for loops 40FC22A−D 18 January.

ρ_3 of 0.02, hence not being regarded as sticky.

Reducing the available length of input vectors to the Yamashita method (read the figures from right to left) does not have a large impact on the calculated indices, as seen from the figure. All the plots are more or less horizontal, indicating that $\partial\rho_{1,3}/\partial$(samples) is small. Down to about 1000 samples there is not much change for these loops, reducing the number of samples even more does introduce some deviation from the "steady-state" of maximum samples.

2.2.4 Noise level in the data

In the Yamashita method for stiction detection one uses the derivative for establishing the symbolic representation of the signals. Due to the derivative, the method will be sensitive to noise. An important question to answer is how the typical noise-level in industrial data affects the method.

For the four loops 40FC22A−D we have recorded data for 6 different days in total, each collection separated by at least a week. By inspection of the data it was evident that some external source was influencing the measurements for all the loops at the same time. For one of the days the signals are plotted in figure 2.10. I don't know this for sure, but I assume that these loops are located close to each other, given their names. Table 2.5 shows the corresponding Yamashita indices and the result of running PCU on the data. By running the software oscillations

Table 2.5: Results of Yamashita method and PCU on loops 40FC22A-D, shown in figure 2.10.

	Yamashita		PCU	
Tag	ρ_1	ρ_3	C.C-B.C-Relay	Verdict
40FC22A	0.08	0.02	–	Non regular disturbance
40FC22B	0.06	0.01	–	Non regular disturbance
40FC22C	0.28	0.12	(I)-S-U	Stiction
40FC22D	0.39	0.14	–	Non regular disturbance

were detected for all the loops. Only for loop 40FC22C did the PCU enter module 3, were it (correctly) indicated stiction in the loop. By inspection of figure 2.10 it seems like this is the loop least affected by the noise. By looking at data from another week, where we did not have the presence of the external noise, it was detected by both Yamashita method and PCU that stiction was present in all these loops.

The important observation from our point of view with these results is that when there is some noise that makes the signal distorted like shown in figure 2.10, the ρ_3 index is low, hence no stiction is detected. This is an important observation, because it implies that it is not "dangerous" to run the method on bad data, it will simply not detect the stiction. Hence the method is robust with respect to noise. This agrees with the simulations conducted. We also see that the more advanced method does not capture the stiction in the valves when the signal is distorted, but it can manage to indicate that there is some external noise. (The non-regular disturbance, which in this case was measurement noise.)

By considering 20 loops which we had data sets for 6 different days, with an interval between the data sets of at least 7 days, we got the idea to use the Yamashita method as a Hägglund index, that is, an index for which at every time the ρ_3 is greater than some given threshold, a counter is increased. This is a robust way of implementing the method, and it has at least these benefits:

- For a loop that is severely troubled by stiction the counter will always increase and hence it will be detected easily.
- If there is trouble with measurement noise, such as discussed above, the method will "wait for" a good signal and then check if there is stiction or not. As long as the signal is bad we expect the ρ_3 to be low, which is a robust feature of the method and this kind of implementation. Further, when the signal is good and there is stiction in the loop, it will be detected.

In figure 2.11 we see an example of how the proposed approach by counting sticky periods might work in practice. If $\rho_3 > 0.25$ is the threshold for increasing

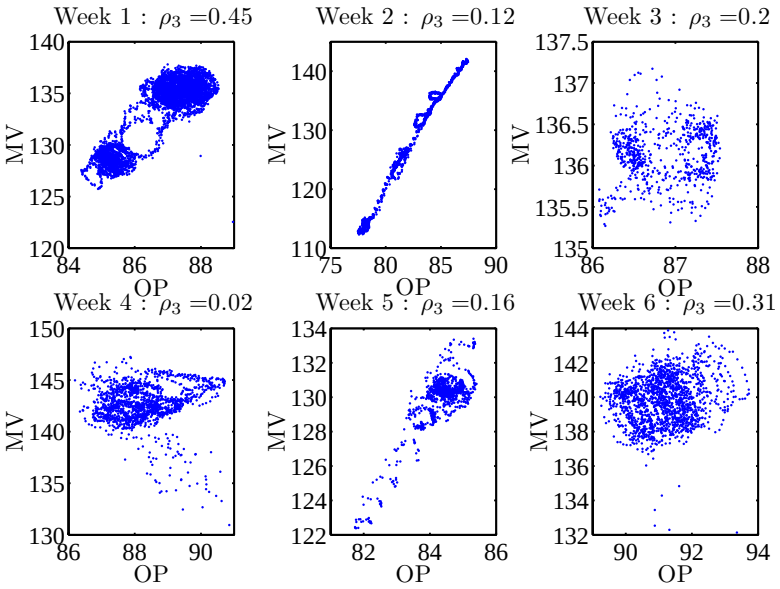

Figure 2.11: The development of loop 40FC22A with time.

the counter for this loop, we observe that in the period of 6 recorded data sets 2 were considered to be sticky, giving an "overall" index of $2/6 = 0.33$. By using visual inspection of the figure (or better by viewing the development of data with time on the computer) we see that for weeks 1 and 6 there is clear signs of stiction in the loop. For week 2 the valve system behaves good, though possibly with some minor occurrence of stiction. For weeks 3,4 and 5 there might be stiction, but we also observe that the noise level is quite high in this case. As previously discussed, by inspecting the MV and OP plots for loops 40FC22A-D for these weeks it was likely that some external source were interfering the measurements, hence making analysis difficult.

By still looking at figure 2.11 it is evident then when the valve is following large set-point changes (for example week 2) the behavior is rather good. When the set point is more constant, for example week 6, the stiction pattern is more evident. What is missing here is obviously some measure of the quantification. Looking at the OP-axes, it seems like the deadband and stickband (S) is the same for all the weeks, hinting that using the span in OP for the stiction pattern can be a good choice for the quantification.

Further, for week 2, when the valve is moving a lot, for large periods of time its velocity is larger than the threshold it needs to go below to stick. When it sticks the pattern is evident.

A simple quantification of noise is to divide the apparent amplitude of the

noise by the amplitude of the underlaying signal. Doing this, for the loops where the Yam method detects stiction, a typical noise level was about 0.1. This level of noise is typically a bit smaller than the noise-level used for simulation. (See table 2.1).

2.2.5 Varying set-points

For a significant number of the loops in which stiction were reported the set point was changing with time. As an example we can look at figure 2.12, which shows a plot of set-point (SP), flowrate (PV) and controller output (OP) for loop 40FC22A, the 15th of February. For this loop the indices were $\{\rho_1, \rho_3\} = \{0.57, 0.31\}$. The PCU characterized the loop as being sticky. One observes that the set point is changing rather rapidly, and we also observe the stiction pattern, especially when the OP is decreasing (PV almost constant in some cases). This is not a rigid analysis, but simply an example where the method does work even though there are set point changes present. Judging from the appearance of the SP-data, this loops seems to be a slave loop.

The period of set point changes in figure 2.12 is about 30 samples, corresponding to a frequency of $\omega = 2\pi/(30 \cdot T_s) = 2\pi/(30 \cdot 10) \approx 0.021$ radians/second. By inspection of the other loops for which the Yamashita method detected stiction and there were more or less linear set-point changes present, it was found that the frequencies of set point changes were about 0.021 radians/second or less. So, from this data it seems like frequencies of 0.02 radians/second or less causes only minor problems for the method. It is difficult to draw exact connections with figure 2.6, because in this case we do not have specific knowledge about the process in question. However, as seen from figure 2.6 we expect it to be more difficult to detect stiction when the frequencies of set point changes increases. (If we assume that the flow loop has a response time of 2 seconds, $\omega/\tau \approx 0.01 = 10^{-2}$. The process in figure 2.6 had $\tau = 10$, hence normalizing the scale by 10 and comparing with the present results shows that the frequency of oscillation is rather high, but perhaps not too high. If the simulation results are trust-worthy, increasing the frequency of set point changes slightly should cause ρ_3 to be lower and stiction should not be detected.) To repeat the conclusion from the discussion performed on the modeled data, given a time-scale separation of 5 or more between the layers, the set point should change in a frequency so low that detection of stiction should not be much affected by eventual set point changes. This agrees with observations for the industrial data; linear-type changing set-points did not seem to cause significant problems. The expected bandwidth for a well-tuned controller should be about $1/\theta^{\text{eff}} \approx 1\ \text{s}^{-1}$, hence the observed changes in set points should be inside this assumed bandwidth.

If the set point changes in a step-wise fashion it is expected to be difficult to

detect stiction, because the steps will (assuming we have tightly tuned controllers) cause the OP and MV to change in a stepwise fashion also. The differentials of the steps will be quite large, and they can dominate the normalization of the differentials (basis for calculation of the indices). This will "hide" the possible stiction pattern and hence stiction might not be detected. As an illustrative example we can look at figure 2.13. One observes that the set point is changing with time, with a step-change at $t \approx 2900$. By running the Yamashita method on all the data $\{\rho_1, \rho_3\} = \{0.36, 0.25\}$, while running the Yamashita method only on the samples after the step (from sample 3000 to the end) resulted in $\{\rho_1, \rho_3\} = \{0.41, 0.29\}$. This set point data had only one pure step. We expect that more steps might introduce more problems and may cause the method to not detect stiction properly.[2] The loop in question was characterized as being under subject of stiction by the PCU.

It should be noted that there where loops in which stiction was detected and the set point was changing in a step-wise fashion. The discussion above shows that a step-wise step change may cause problems, but with severe stiction present the Yamashita method can still detect it. Mathematically this depends on the transfer function from the set point to controller output and flowrate, relative size of steps to size of jump and number of steps conducted.

As a conclusion, this small analysis suggests that set point changes with a smooth fashion causes only minor problems for detection of stiction, whereas step-wise changes with a not too high frequency in the set point *may* cause more difficult problems.

As seen in the example (and confirmed by simulations), step-changes in the step-point causes the indices to *decrease*, which implies that it does make it more difficult to detect stiction, but we will not wrongly detect stiction, which could be the case if the indices increased with the step-changes.

2.2.6 The matched stiction index ρ_3

In his paper Yamashita writes that for a random signal the index ρ_1 should have an index about 0.25 for a completely random signal. For 210 loops investigated, including pressure and temperature loops (which may be regarded as random since we know that the method will not work for these loops), gives a mean of 0.252 (!). Further we know that $\rho_3 \leq \rho_1$, where the equality holds for a perfect stiction pattern.

Figure 2.14 shows how the index was corrected from ρ_1 to ρ_3 by calculating

[2] Adding more steps will make the standard deviation of the differential increase, since the differential of a step is very high. This will lead to an increased threshold in the identification, where a too large threshold will cause stiction patterns to be regarded as "SS" by the method, because in the extreme case only the steps will be regarded as either increasing or decreasing.

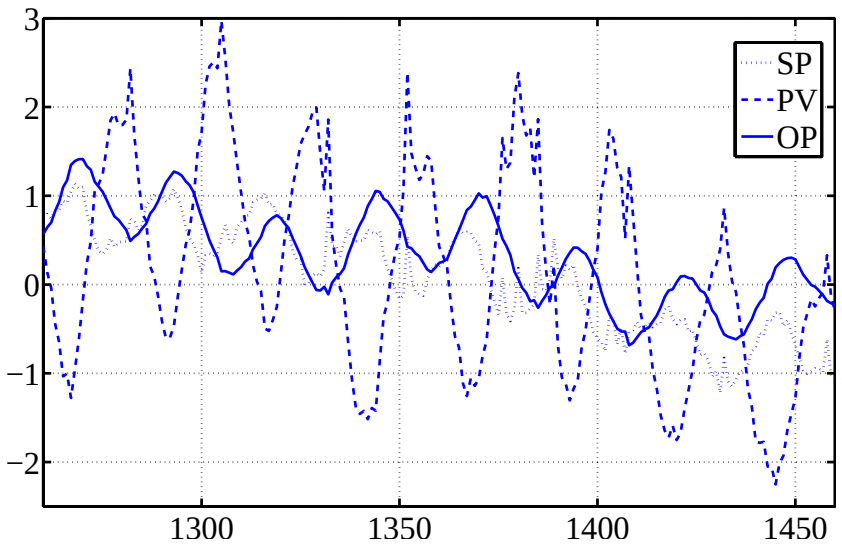

Figure 2.12: Some of the samples for loop 40FC22A, 15. February. We had 1615 samples in total. The mean is subtracted from all the signals.

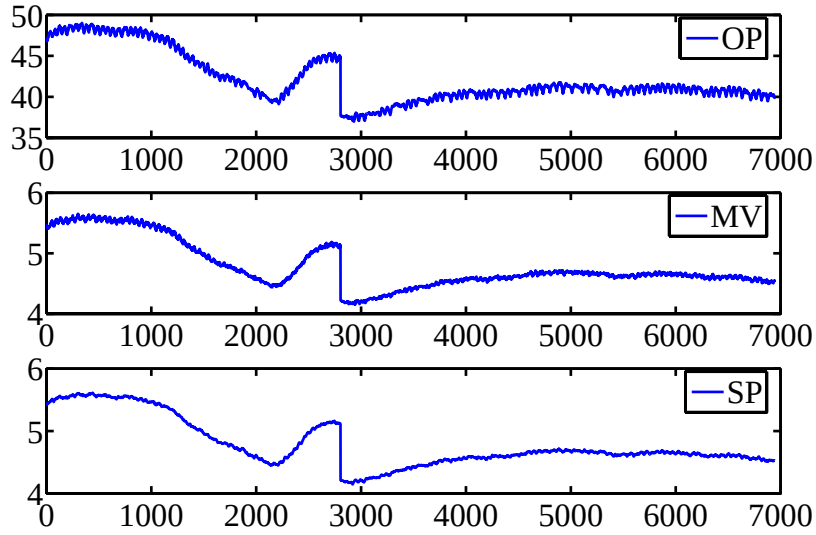

Figure 2.13: Recorded data for loop 40FC95, 13 December 2004.

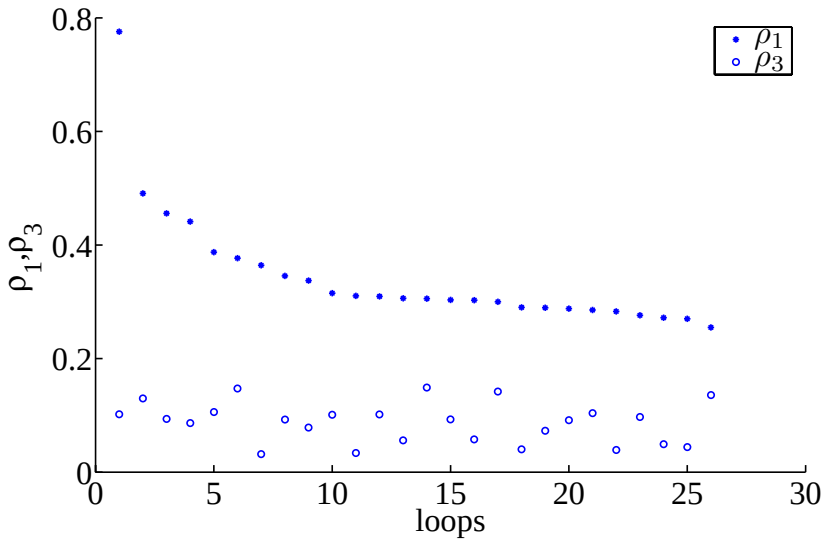

Figure 2.14: Some loops where the calculation of ρ_3 was important.

the matched stiction index ρ_3. For these 26 loops the ρ_1 was above 0.25, while the corrected index ρ_3 was below 0.15. This illustrates the importance of calculating ρ_3, and this index should be interpreted as the "true" Yamashita index.

2.2.7 Industrial implementation

The Yamashita stiction detection method is rather fast. To analyze one loop with 1600 samples takes about 0.1-0.2 seconds with a naive implementation of the routine in Matlab.[3] (See appendix A for details.) Figure 2.15 shows the architecture of the current control loop performance monitoring system (CLPM) with the PCU being located on an external computer (Advanced Layer). See [23] for details. The idea of the current architecture is that a (modified) Hägglund index sorts of the oscillating loops and sends the interesting data further to the external computer with PCU implemented. Ideally the Hägglund index is implemented within the DCS. However, due to restrictions set by the refinery[4], for the time being also the Hägglund index is in a layer of it's own. As this is under development, we have chosen to label this layer with an X. This layer contains interfaces to exchange data between DCS and the advanced layer.

With the observed properties of the Yamashita method, we propose three ways of implementing the method. The first two are implementations integrated within

[3] On a Pentium 4, 3 GHz, with 1 GB RAM.

[4] The PCU is currently under testing on an Italian refinery

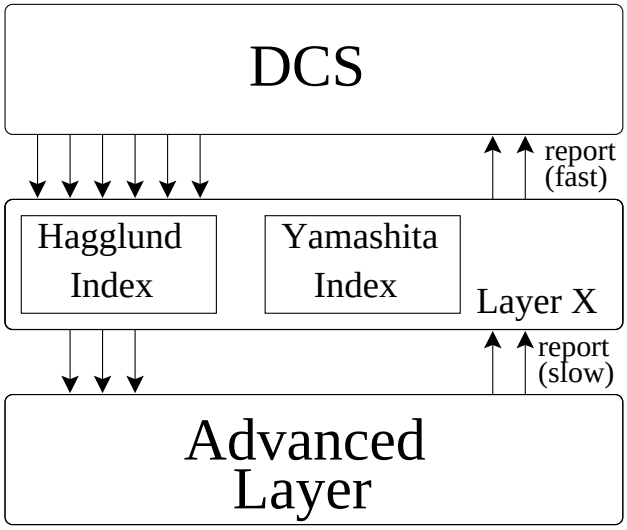

Figure 2.15: A possible implementation of the Yamashita method with the current DCS system.

the PCU, first in series and then in parallel. The third is as a stand-alone application.

Integrated in advanced software package, series We have observed that the Yamashita method detects the loops which are clearly affected by stiction with a typical industrial noise-level. A possible implementation together with the current software could be to implement the Yamashita method together with the Hägglund method in the X-layer as indicated in figure 2.15. An economic structure could be to first use the Hägglund index to find oscillating loops, then pass the oscillating loops to the Yamashita method. If the Yamashita method detects stiction, some alarm (or counter) could be sent (increased), while in the cases where the Yamashita method does not detect stiction we send the data to the more advanced and time consuming software located in the "advanced layer".

Integrated in advanced software package, parallel Another possible implementation together with the PCU is to include the method in parallel with the other methods. Today three methods for stiction detection is used, and if 2 of them reports stiction internally, PCU reports stiction externally. Since the Yam method has proved to be good, it can be implemented as a fourth method here, for example requiring in PCU that at least $3/4$ methods needs to report stiction to achieve an overall "stiction" verdict.

Stand-alone version As discussed in the sections above, given that a loop that behaves good (or non-sticky) will have a low ρ_3, we can consider making a stand-alone Yamashita block, where we would have the method itself and some counter in analogy to the Hägglund index. When the ρ_3 exceeds some given limit the counter increases. When the counter is higher than some given limit (maybe scaled in time, that is, numbers of high ρ_3 per time unit), the block sends a report/alarm back to the DCS system. This alarm would then be passed on the the operators.

Remark 1 *For the option "Integrated in advanced software package, series", one could of course direct the data flow directely to the Yamashita block, hence there are really two possible configurations for this options. However, strictly speaking this would be the same as running the "stand-alone verison" in parallell with the PCU. Of course they could "communicate", for instance if there is obvious stiction found with the Yam method one does not need to initate analysis by the PCU. This discussion shows that for the final industrial implementation some more thought and experimentation needs to be done. In the paper submitted to ANIPLA we suggested to let the Yam method report the loops with obvious stiction, leaving the other oscillating loops to closer analysis by the more advanced methods.*

Remark 2 *For an industrial implementation it could further be worthwhile discussion if an on-line implementation is necessary, or if it is good enough to send data to an external computer and do analysis off-line at a regular basis.*

2.2.8 Saturation of valve

It is rather obvious that when the valve saturates this can wrongly be detected as stiction, since the valve is not moving. This should be easy to fix by including the range of the flowrate in the implementation. An example where stiction was wrongly indicated is shown in figure 2.16. (This is from another data set than the other loops discussed in this chapter. For local reference this is from the "Sartec" data.) Here $\rho_3 = 0.27 > 0.25$ when applying the method on all the data including the saturation of the valve. Running the method on data before and after the saturation gave $\rho_3 = \{0.18, 0.20\}$, hence not indication of stiction in the loop.

Often the controller output OP saturates. Theoretically this should not introduce problems by wrong identification, because it yields primers on the form τ_{Sx}, where $x \in \{\text{I, S, D}\}$, hence not being regarded as stiction in calculation of ρ_1.

After some further thought one realizes that for the problem stated above to be a real problem the anti-windup in the valve must not be working properly. If the

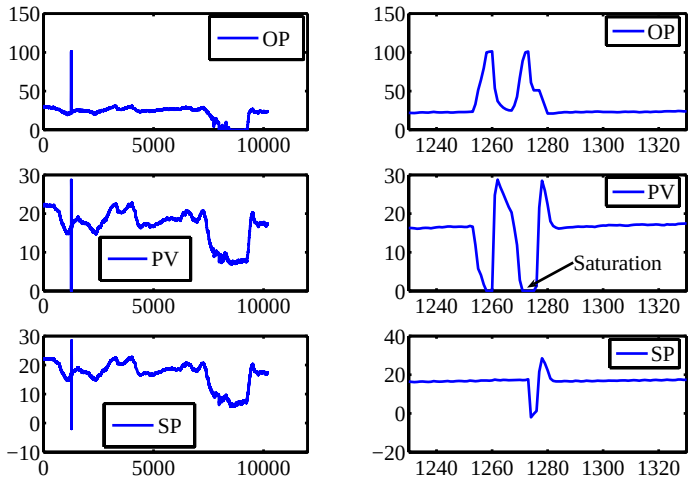

Figure 2.16: Example of saturation of the valve. From industrial data.

anti-windup is working correctly both valve and controller output signal should be steady when the valve saturates. This is a tricky case that we need to take care of. Consider the calculation of ρ_1:

$$\rho_1 = (\tau_{\text{IS}} + \tau_{\text{DS}}) / (\tau_{\text{total}} - \tau_{\text{SS}})$$

In the case when both signals is steady during the period of investigation, $\rho_1 = 0/0$, so this calls for another `if` statement in the implementation. During the investigation of the industrial data no example of this situation was found, but theoretically it is a possible problem.

2.3 Summary and conclusions

In this chapter the Yam method has been tested on both simulated data generated by the Choudhury model and plant data from Italian plants. The method has only been applied to flow loops from plant data, as it is only valid for these loops. Of 216 loops inspected 167 were flow loops, which is about 3/4 of the total loops.

From the investigation of several points of interest in the light of industrial applications, the following conclusions can be drawn:

- The method is based on derivatives and uses the standard deviation as threshold for the symbolic identification of stiction pattern: some sensitivity to the level of noise is shown in simulation. The low level of noise encountered in most of industrial registrations makes this drawback less relevant.

- The observation window can be limited to few periods of oscillations and, for sampling time $T_s = 10$ seconds, a default of 2 hours can be suggested.

From the comparison of results on industrial data with a package which performs a sequential application of stiction detection methods, it can be concluded that stiction is recognized in about 50% of cases. Therefore the Yam technique can be suggested for a fast identification of sticky loops with clear patterns, leaving more difficult loops for a deeper analysis, with two advantages: a quick detection of the onset of stiction and a save of computation time. This characteristics, together with the ease of implementation of the algorithm with any simple language, makes the technique suitable for on line implementation.

The next chapter will include some more discussion about why the method did not work in some cases, and some other patterns found in industrial data and comparison with control equations to explain the patterns found.

Chapter 3

Patterns found in loops suffering from stiction

While carrying out the work using the Yam method to industrial data, a lot of different patterns were observed. The Yam method is based on how the plot evolves with time. I therefore found it useful to make the plot evolve on the computer screen to also see the development with time. This was a fundamental tool for the work done as a basis for this chapter. This "movie" function is enabled in the GUI, see appendix A.2 for more details.

As more patterns were discovered I wanted to find the physical reason for the patterns observed. Since flow-loops with incompressible fluids were investigated, I thought this should be possible with using simple equations learned in basic control courses.

This analysis has been the basis for some discussions at the lab during spring 2006, I hope this chapter can be clarifying and maybe introduce new ideas.

The notation used in this chapter follows logically from the previous chapters. For example, if I write IS, this means a time-period (of one sampling interval) where the controller is increasing its output (I) while the valve-position or flow-rate is steady (S).

3.1 The PID controller working on a sticky valve

The ideal form of a PID controller without a derivative filter and its derivative are:

$$u^{\mathrm{PID}}(t) = K_c\left(e(t) + \frac{1}{\tau_I}\int_0^{\tau} e(\tau)\mathrm{d}\tau + \tau_D\frac{\partial e(t)}{\partial t}\right) \tag{3.1}$$

$$\frac{\partial u^{\mathrm{PID}}}{\partial t} = K_c\left(\frac{\partial e(t)}{\partial t} + \tau_D\frac{1}{\tau_I}e(t) + \frac{\partial^2 e(t)}{\partial t^2}\right) \tag{3.2}$$

Usually the derivative part is filtered, but if this is the case or not is not really important here.

First of all, when the valve is stuck $\partial e(t)/\partial t = \partial^2 e(t)/\partial t^2 = 0$, so the integral action is fundamental to get the IS and DS periods. For example a pure P-controller will just keep its output constant and there will be an SS situation when the valve is stuck.

From equation (3.2) we observe that the sign of the derivative of the control signal for a PID controller is:

$$\begin{aligned} \operatorname{sign}\left(\frac{\partial u^{\mathrm{PID}}}{\partial t}\right) = \\ = \operatorname{sign}\left(K_c\left(\frac{\partial e(t)}{\partial t} + \frac{1}{\tau_I}e(t) + \tau_D\frac{\partial^2 e(t)}{\partial t^2}\right)\right)\Bigg|_{\partial e(t)/\partial t = \partial^2 e(t)/\partial t^2 = 0} = \\ = \operatorname{sign}\left(K_c \cdot \frac{1}{\tau_I}e(t)\right) \end{aligned} \tag{3.3}$$

The "overall gain" for controller-valve-plant should always be positive if we assume negative feedback and require stability. Focusing on flow loops, the plant gain $|G|$ can be assumed to be positive. In the valve there are usually two situations:

- Direct action: When the control signal increases the valve opens more. Hence the gain is positive.
- Reverse action: When the control signal increases the valve closes. This gives a negative valve gain.

For a direct acting (DA) valve in a flow loop the controller gain K_c is required to be positive for stability, while for a valve with reverse action (RA) K_c is required to be negative. This further gives rise to two different patterns in the MV(OP) plot. For the DA case, when $e(t) > 0$ the sign of change of OP is positive, since K_c is positive, while for the RA case the sign of change of OP is negative when $e(t) > 0$.

This implies that the expected directions in a MV(OP) plot is counter clockwise for the DA case, and clockwise for the RA case. The Yam method as proposed in [34] is designed to detect stiction for DA-valves. If a RA-valve should suffer from stiction, the ρ_1 index should be high, whereas $\rho_3 \to 0$, because all the sticky patterns are removed in the calculation of this index as they do not match the expected DA counter clockwise pattern. This is illustrated in figure 3.1. The dot in the figure is the set point for the valve and controller.

If the clock-wise RA plot (right hand side of figure 3.1) is reflected around the OP axis the pattern will be a counter clockwise plot exactly matching the DA plot.

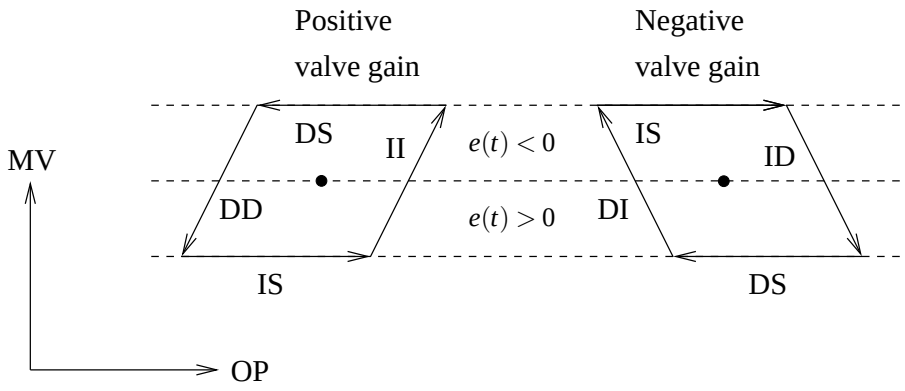

Figure 3.1: Positive and negative valve gain.

Hence, if it is known that the valve has reverse action one can use (OP,−MV) to calculate ρ_3.

The reason for initiating the discussion above was plots of MV(OP) as shown in figure 3.2. The plot available at first was the right-hand-side plot without the arrows that were added manually later. This resembles a reverse acting valve as the plot bends to the left. By inspecting the data and its evolvement in time, the direction was found to be as indicated by the arrows. In the left part of figure 3.2 the number of samples has been given its own axes to try to visualize the evolvement on paper. However, this is rather difficult without the added arrows. Using a computer makes this visualization easy, but still a method for putting it on paper is lacking. The arrows does the job nicely, but I still haven't found a way to do this automatically in Matlab.

The visualization was used on all the loops were stiction was indicated, and all of them had a counter clockwise plot. This implies that all the valves in the plant data were of the direct action type. We expected this number to be about half, since by design about half of the valve should be reverse acting. After a correspondence with Yamashita, he suggested that in modern valves there is always a positioner. A sketch of a valve is shown in figure 3.3, with the positioner in place. If the action of the valve is reverse, usually also the positioner is set in reverse, hence making the whole system direct. The sign in the positioners for the data studied in this report is not known.

3.2 Closer look at the control equation

Why it is so that the loop in figure 3.2 and also other loops had a plot that was bending to the left, if it was not reverse action? Flow loops have fast dynamics

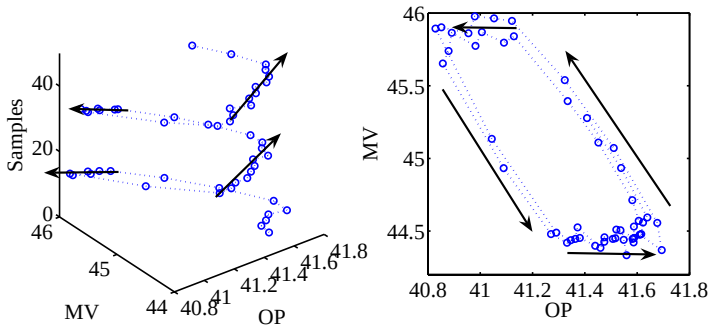

Figure 3.2: Example of apparent negative valve gain. With the arrows ones sees that the gain in the valve must be positive, as the plot is counter clockwise. However, without the arrows deciding this is difficult. In the left figure time (samples) is given its own axes, but without the added arrows it's still difficult to see the rotation of the plot.

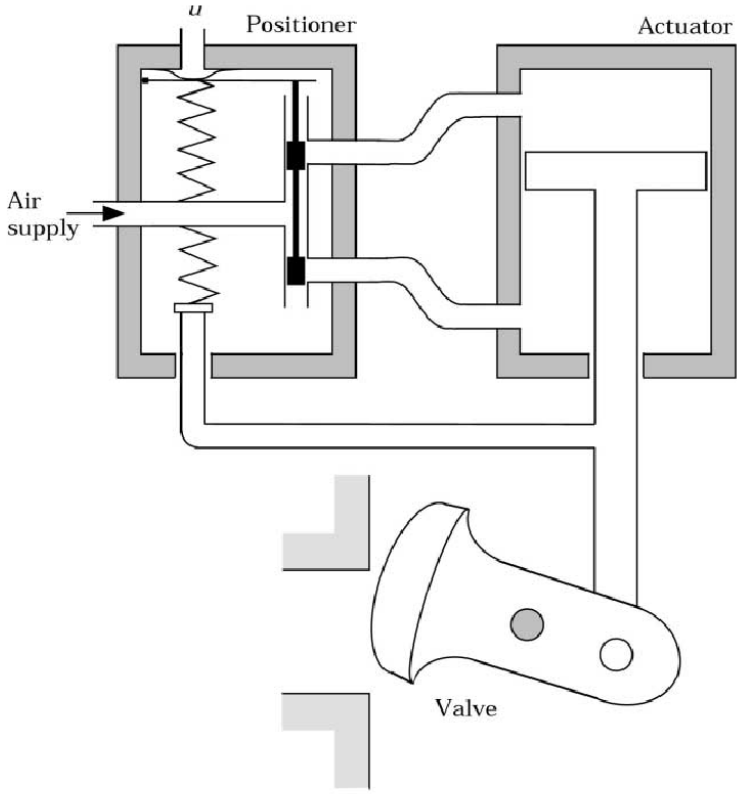

Figure 3.3: Sketch of valve with positioner. Taken from [8].

and therefore it should be possible to explain this phenomena using the basic equations.

Assuming that the flow-loop can be modeled as a first order plus delay system, the process transfer function is $g(s) = k\frac{e^{-\theta s}}{\tau s+1}$, while a PI controller has the equation $u(t) = K_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(\tau)d\tau\right)$ in time domain. It is assumed that measured flow-rate y_m is proportional to the valve position MV. This assumption implies that $g(s) \approx 1$ for the frequencies of interest. It further implies that $\tau s \approx 0$ and $\theta s \approx 0$. For a valve suffering from nonlinearities the deviation normal linear operation can introduce a pattern in the MV(OP) (or y_m(OP)) plot which in turn should be captured by the pattern recognition techniques.

In common for all the pattern is that they have a period when the valve is stuck (IS) and then a period when the valve is moving. When the valve starts to move, all the patterns have also in common that the flow-rate is increasing. (We are now considering the IS-parts if the patterns, the DS-parts is just a mirror of the IS-parts.) The controller output is either increasing (I), steady (S), or decreasing (D) after the have slipped. Let us consider the derivative of the controller output:

$$u(t) = K_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(\tau)d\tau\right) \tag{3.4}$$

$$\frac{\partial u}{\partial t} = K_c\left(\frac{\partial e(t)}{\partial t} + \frac{1}{\tau_I}e(t)\right) \tag{3.5}$$

$$\frac{1}{K_c}\left|\frac{\partial u}{\partial t}\right| \leq \left|\frac{\partial e(t)}{\partial t}\right| + \frac{1}{\tau_I}|e(t)|, \tag{3.6}$$

where the last equation is the triangle inequality.

For a constant set point the error $e(t) = r(t) - y_m(t) = -y_m(t)$. For the time when the controller was integrating while the controller was stuck, $y_m(t) < 0$ (let all the variables be deviation variables from now on). To clarify, inserting $e(t) = -y_m(t)$ in the control equation yields:

$$\frac{1}{K_c}\frac{\partial u}{\partial t} = -\underbrace{\frac{\partial y_m(t)}{\partial t}}_{>0} - \frac{1}{\tau_I}\underbrace{y_m(t)}_{<0} = \tag{3.7}$$

$$= -\left|\frac{\partial y_m(t)}{\partial t}\right| + \frac{1}{\tau_I}|y_m(t)| \tag{3.8}$$

Note that *the last equality holds only when the valve position is increasing from under its setpoint towards its setpoint*, i.e. only in the case considered here. To be more general the ususal triangle inequality should be used here, but in this special case we can use the equality directly.

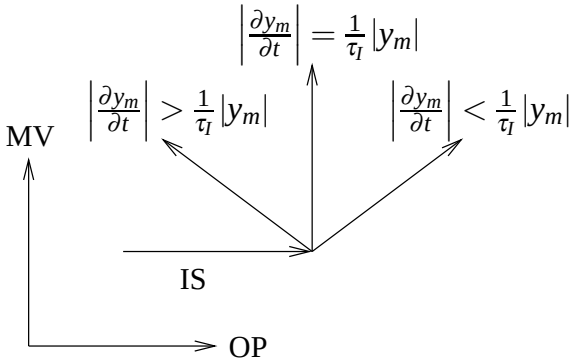

Figure 3.4: Possible directions of the slip-jump after an IS period.

Considering (3.7) one can explain all the patterns, because we have opposing factors. When the valve jumps, making y_m *increase* from its negative value the first term is positive, while the second term will be negative as long as y_m stays belove zero.

To clearify further what this means for implementation of the Yam method, consider figure 1.7, page 17. Yamashita writes that these two patterns are the expected patterns in sticky valves, based on experience. However, depedning on the relative sizes of the terms in equation (3.8), another pattern is likely to appear in sticky valves. This is illustrated in figure 3.4. Aften one (or more) IS period(s) the controller/valve can theoretically move in 3 directions when the valve starts moving. The cases of II and SI are expected by Yamashita and also agrees with figure 1.2, page 5, which is taken from [4]. However, as seen from the above derivation the valve can also move "to the left" if $\left|\frac{\partial y_m}{\partial t}\right| > \frac{1}{\tau_I}|y_m|$.

Of course, ρ_3 can be modified to include this "new pattern", or a new ρ_3 for the new pattern can be made. The latter suggestion seems most robust, as we want to withdraw a lot of patterns from ρ_1 to make ρ_3 as sharp as possible.

3.3 Delay in measurements

Even though the dynamics in flow-loops should be negible, there may be delays in measurements, or "measurements out of phase" with each other. The effect of this is rather easy to simulate in Matlab when plant data is already avaliable. Assuming that there are recorded data $\text{MV} = (y_1, y_2, \ldots, y_N)$ and $\text{OP} = (x_1, x_2, \ldots, x_N)$ avaliable. If we want to investigate a delayed measurement in MV, we simply shift the vector with a fixed number of steps. For instance, if the wanted delay is 4 time-units, $\text{MV}^{\text{delayed}} = (y_i^{\text{delayed}})_{i=1}^{N-4} = (y_j)_{j=5}^{N}$. The effects of delays in both

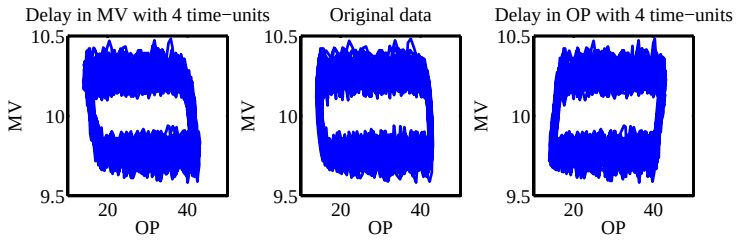

Figure 3.5: Effect of measurement delay in MV and OP.

MV and OP with 4 time-units is illustrated in figure 3.5. Here simply the data vectors were shifted as indicated over. The original data had a vertical jump in the MV-data.

It is evident that measurement delay (in this case maybe unrealisticly high, 4 time units is 40 seconds in this data set) may make the plot bend both to left and right.

Remark *The "Delay explanation" for why the plot may bend to the left seems rather unrealistic as we needed 40 seconds delay to get clear results. Personally I have stronger belives in the explanation given in the previous subsection, displayed grahpically in figure 3.4.*

3.4 Strong increase followed by weak increase in the OP signal

For two loops obviously suffering from stiction the simple index ρ_1 was low, of course making ρ_3 low afterward. For these two loops the OP signal was characterized by first a strong increase and then a weaker one. An example is shown in figure 3.6. In the figure also the normalized derivatives are plotted. From the derivative we see that the Yamashita method only detects the first strong increase as "increase (I)", the weaker increase is symbolized as "steady (S)". This is the reason why the method fails in this case. Remember that the standard deviation of

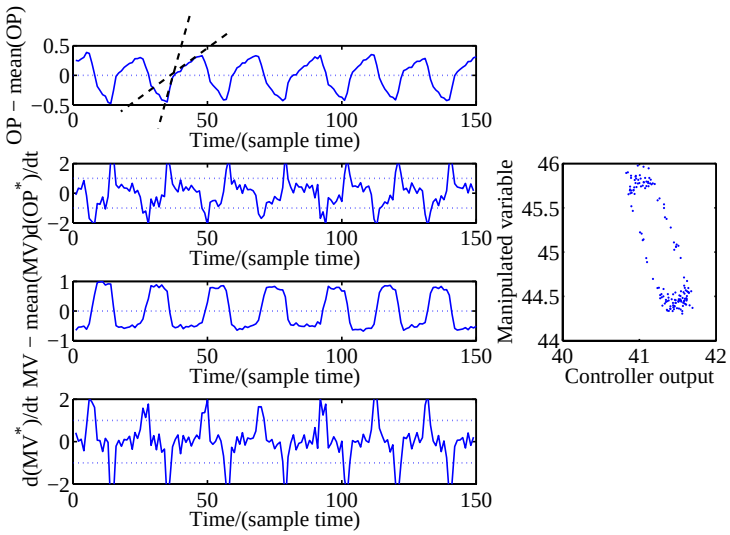

Figure 3.6: Example of OP signal with first a strong and then a weaker increase in a loop affected by stiction.

the differentials are used as a threshold in the identification. This is the horizontal dotted line ($=1$) for the normalized differentials $\mathrm{dOP}^*/\mathrm{d}t$ and $\mathrm{dMV}^*/\mathrm{d}t$. For a recapture of the normalization see page 15.

The pattern can also be explained by taking a closer look at the recorded signals as seen in figure 3.6 and considering a typical PI controller with its time-derivative:

$$u(t) = K_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(\tau)\mathrm{d}\tau\right)$$
$$\frac{\partial u}{\partial t} = K_c\left(\frac{\partial e(t)}{\partial t} + \frac{1}{\tau_I}e(t)\right)$$

By inspection of the plot we observe that after the periods of the valve being stuck it does not jump directly to its new position, it travels a bit and then sticks again. This is also evident by looking at the OP−MV-plot, where we see that there are some points also in the travel phase. When the valve jumps/starts traveling, the change in error $\partial(e(t))/\partial t$ will be large and as seen from the control equation above this will make $\partial u/\partial t$ large. Hence, as also seen from the figure, the travel-phase of the pattern is the phase in which the controller output is symbolized (correctly) as increasing or decreasing. When the valve is stuck and only the integral action in the controller is working ($\partial(e(t))/\partial t = 0$) the change in controller output is less ($K_c \cdot e(t)$) and hence this period is symbolized as being “SS”. I have

no suggestion for how we can improve the method to detect also these cases at the time of writing, but I thought this was an interesting observation.[1]

The same kind of trend of the OP signal can be found by looking at figure 4 (a) in Rossi and Scali [20]. For the case of significantly delay ($\theta/\tau = 10$) their simulation gave the same result as the real data of loop 51FC102 (The loop in figure 3.6). A delay-dominant process gives band-width limitations for stable tuning of the controller, and we expect that the controller is tuned rather slow. Now considering figure 4 (c) in [20], it seems like lowering the controller gain can introduce the same pattern in OP, though to see this more clearly they should have made simulations with even smaller controller gain.

A level loop can be tuned either tight or smooth, often it is desirable to tune it smooth to use the filter-properties of the tank, see for example [24]. If the observations and implications outlined above are true, this implies that for averaging level tanks the typical observed pattern with stiction of the valve will be of a kind that the Yamashita method can not detect with it's simple implementation as used in this report.

Yamashita writes that the method can be extended to detect any known pattern, but in this case I think it can be difficult to extend it easily.

3.5 Summary and conclusions

This chapter has been like a walk with the PI(D) control equation and how it works on valves suffering from stiction. Some of the cases where the Yam method did not detect stiction have been explained.

Some of the "new" patterns, for instance when the stiction pattern that is bending to the left, can be included in the Yam method by modifying the ρ_3 index. After the industrial applications it was evident that to get a clear indication we need to withdraw quite a number of patterns from $\rho_1 \rightarrow \rho_3$. Therefore, I think the "correct" way to include the new patterns is to make "new" ρ_3's, instead of including more patterns allowed in the original one. However, this may be seen as a tuning of the method.

After my leave from CPC-Lab there will be initiated a cooperation with Enel. Enel is Italy's biggest power company, and Europe's second-largest listed utility. (Taken from www.enel.it). In their lab at Pisa they have a valve in which the amount of stiction can be varied. Next students' projects will be to conduct experimental work on this valve. Hopefully then the patterns described theoretically in this chapter can be verified. This work has been on a qualitative level, missing numbers and information to make it quantitative.

[1] Local footnote: Another loop in which the same pattern is observed is loop FC445 from the Horch data.

Chapter 4

Quantification of stiction

Stiction detection methods such as the bi-coherence [4] and relay [20] can inherently only indicate the *presence* of stiction, not the *magnitude*. Figure 4.1 shows an example of two loops with stiction detected by the Yamashita method, with ρ_3 indices similar. The magnitude of stiction is yet to be defined, but using for example the change in controller output when we valve is sticky as a temporary definition, clearly there is a significant difference in the stiction magnitudes. This calls for a way to asses the magnitude of stiction.

Choudhury et al. [4] proposes a method which will be reviewed later in this chapter. Rossi et al. [21] proposes a method based on the magnitude of the controller output signal when the valve is stuck and normalizes with the span of the output. This method is already investigated at the lab. The method has shown to be promising by application on historical plant data, but it has yet to be implemented on-line.

4.1 Stiction quantification by fitting an ellipse

This method is based on the work by Choudhury et al. and presented in [4]. The basic idea is to plot OP and MV in the usual plot and fit an ellipse to the data. Then stiction is quantified by using the size of the fitted ellipse in OP units. The result should be "detected stiction in percentage of valve travel span". Instead of directly fitting an ellipse to the data some clustering techniques can be applied. This is all described in the reference.

Some filtering of the data is necessary to reduce the influence of noise. The method is connected to the bi-coherence method. The bi-coherence method is based on the fact that to describe the nonlinear behavior of a signal we need to use the bi-spectrum (in frequency domain). The bi-spectrum describes phase coupling between some frequencies f_1 and f_2. For a linear signal this coupling is zero. The

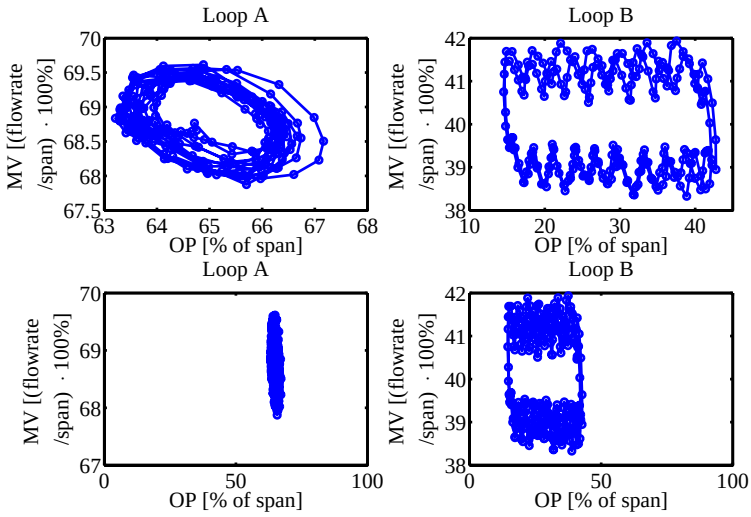

Figure 4.1: Illustration of the need for quantification of stiction. Loop A had $(\rho_1, \rho_3) = (0.36, 0.34)$, whereas loop B had $(\rho_1, \rho_3) = (0.47, 0.38)$.

bi-coherence gives the same information as the bi spectrum but is normalized to a value between 0 and 1. Since stiction is nonlinear, the peaks in the squared bi-coherence plot can be used to find filter boundaries for an approximate Wiener filter.

If was not completely clear how the approximate Wiener filter should be formulated. In [16] a realization is shown, which requires a model of the noise in frequency domain. This is know as a true Wiener filter. An approximate Wiener filter is simply a filter where the power of the unwanted frequency channels is set to zero. A simple implementation is to take the fast Fourier transform (FFT) of the signal, set all unwanted coefficients to zero, and then take the inverse transform to get the filtered signal in time domain.[1] A detailed description and implementation in Matlab are given in appendix C.

One has to be very careful when counting the frequency channels using this method for filtering. That's because the Fourier transform is double-sided (i.e. it includes the aliased frequencies above the Nyquist sampling frequency), so one has to make quite sure that the aliased channels are set to zero too before doing the inverse transform. An easy way to test if the implementation is correct is to look carefully at the final filtered time-domain signal following the inverse Fourier transform step. If one has got it right, then the sizes of any imaginary parts are

[1] Since it was unclear how the filter should be formulated an email was sent to prof. Nina Thornhill, a co-author of [4]. She confirmed that the filter was as simple as this.

negligible and are just due to round-off error. See reference [30] for how to deal with the aliased channels. (Thanks to N. Thornhill for the above recipe.)

The complete stiction detection and quantification scheme as presented in [4] is then:

1. Detection of Nonlinearity. Calculate the indices NGI (NonGaussianity Index) and NLI (NonLinearity Index) for the control error signal $e(t)$[2]. If both these are larger than some threshold values, the loop is detected as nonlinear.

2. Filtering of the PV and OP data.

   (a) Once the nonlinearity is detected, obtain the frequency (f_1', f_2') corresponding to the maximum bi-coherence peak in step 1. Note that all frequencies are normalized such that the sampling frequency is 1.

   (b) The boundaries of a Wiener filter can be obtained from

   $$[\omega_L = \max\left(0.004, f_1' - 0.05\right), \omega_H = \min\left(0.5, f_2' + 0.05\right)].$$

3. Obtain the segment of data with most regular oscillations.

4. Fit an ellipse.

5. Quantify stiction.

For now focus will be on the filtering part. The lower boundary is at minimum 0.004 samples/cycle, or 250 samples/cycle. Any oscillation with less samples per cycle will automatically be filtered out. (However, in their illustrating example, [4, page 8], the lower boundary in the filter is 0.001 samples/cycle, thus a minor contradiction in the paper.) The upper limit is due to the Nyquist sampling theorem, using the FFT one cannot get information about frequencies above $1/(2\Delta)$, where Δ is the sampling interval [16]. Since the problem is normalized with sampling time this upper frequency is $1/2$.

In thought the filter should remove the high-frequency noise and the low-frequency drift. The drift can also be removed by using the Matlab command "detrend", which fits a straight line through the data and removes this line from the points. When working with FFT's in Matlab it is recommended to use "detrend" on the data first.

[2]This is what is known as the bi-coherence method for stiction detection. This method is already implemented in the software package and thus will not be discussed in detail in this report.

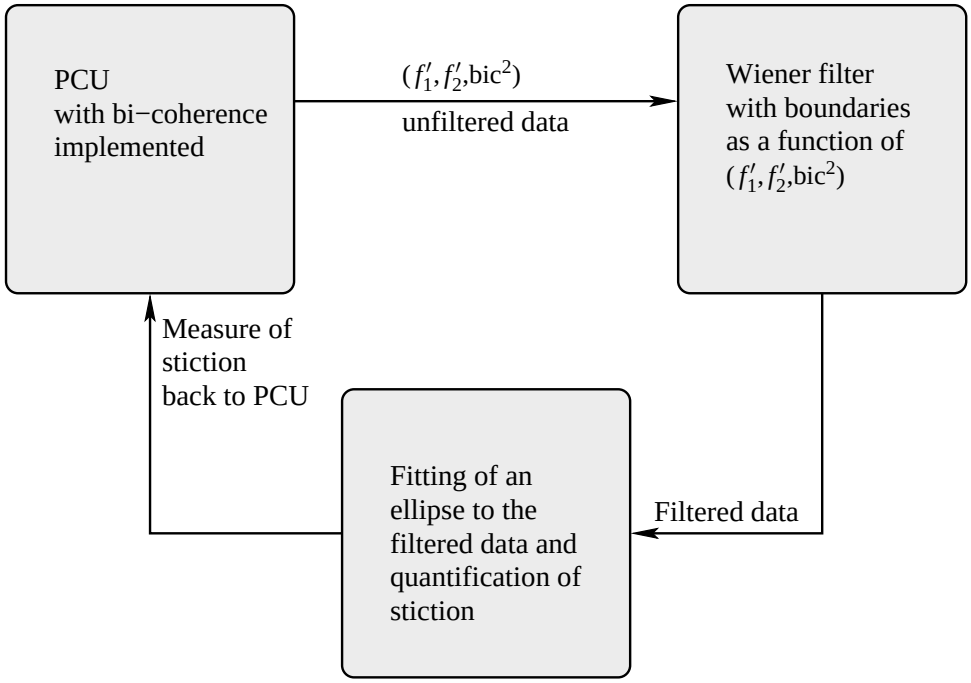

Figure 4.2: Steps necessary to perform the quantification.

4.1.1 Using the filtering technique on two loops with apparent stiction

This subsection contains a test of the filtering described above. Neglecting the selection of the segment of data with most regular oscillations (there was not time to investigate this point), the flow-sheet for calculations is shown in figure 4.2. The next section will focus on the fitting of an ellipse to the data.

The first loop investigated is loop B in figure 4.1. The bi-coherence of the control error $e(t) = y_{\text{sp}} - y_m$ and two possible boundaries for the FFT-based filter are shown in figure 4.3. The highest peak in the bi-coherence plot is $(f_1, f_1, \text{bic}^2) =$ (0.125, 0.109, 0.181), while there is a lower peak at $(f_1, f_1, \text{bic}^2) =$ (0.359, 0.117, 0.164). Using the highest peak as a basis for the filter boundaries, the filter should be

$$\omega_L = \max(0.004, 0.109 - 0.05), \omega_H = \min(0.5, 0.125 + 0.05)$$
$$\omega_L = 0.059, \omega_H = 0.175$$

From figure 4.3 it is evident that the lower filter boundary is too high, as the filter in this case filters out the stiction pattern itself. Using instead the minimum value of 0.004 for the lower boundary conserves the pattern in the recorded signals.

Extending the filter boundaries to also include the other peak does not change the results much. This indicates that the stiction pattern had a low-frequency component that is not shown in the bi-coherence plot of the error signal, since there was a need to lower the boundaries to observe the pattern in the filtered variables.

The other loop studied is also a loop with an apparent clear stiction pattern, but this loop was also subject to frequent set point changes. Its bi-coherence and filtered signals as well at the set point in time domain is plotted in figure 4.4. By close inspection of the set point signal on sees that the period of changes is about 20 samples/cycle, hence a frequency of 0.05. Interestingly, by studying the bi-coherence plot of $e(t)$ closely one sees that there is no significant bi-coherence between this frequency and other frequencies.

For this loop there were only 4000 points stored. Since it is recommended that the analysis should be done on a number that is equal to a power of 2, the 2048 first points were used for computing the bi-coherence plot.

The highest peak is at $(f_1, f_2, \text{bic}^2) = (0.0625, 0.078, 0.76)$, which places the filter on $[0.0125, 0.128]$. The result of the filtering, see figure 4.4, is that the clear stiction pattern in the unfiltered data is changed to a less clear and more round pattern in the filtered data. For clarity only the first 256 points are plotted in these figures.

By still looking at the bi-coherence plot on sees that there is also bi-coherence between $f \approx 0.08$ and other frequencies up to about 0.4. Some experimentation was conducted to find the "tightest" filter boundaries which still reproduces the signal satifactionally. It was found that a filter of $[0.004, 0.4]$ did the job. This means that almost all frequencies were included, which is reasonable given the many high peaks in the bi-coherence.

Ideally the filter should remove the effect of noise and the set point changes, leaving only the clear stiction pattern in the filtered variables. Since the frequency of set point changes is within the bandwidth for the filter, this was not filtered out. For the case of noise, there was not much noise in the original data and hence filtering was not necessary for this purpose.

Discussion

Out of a lot of available plant data the two loops discussed above were selected because the they seemed to be most readily analyzed by the current method. The FFT is generally sensitive to non-stationary data [4]. For most of the flow loops available in the industrial data the set point was changing in a more irregular fashion than for the loop in figure 4.4. Even on these two simple loops it was seen that after using the initial "automatic" settings for the filter boundaries tuning was needed to get good results.

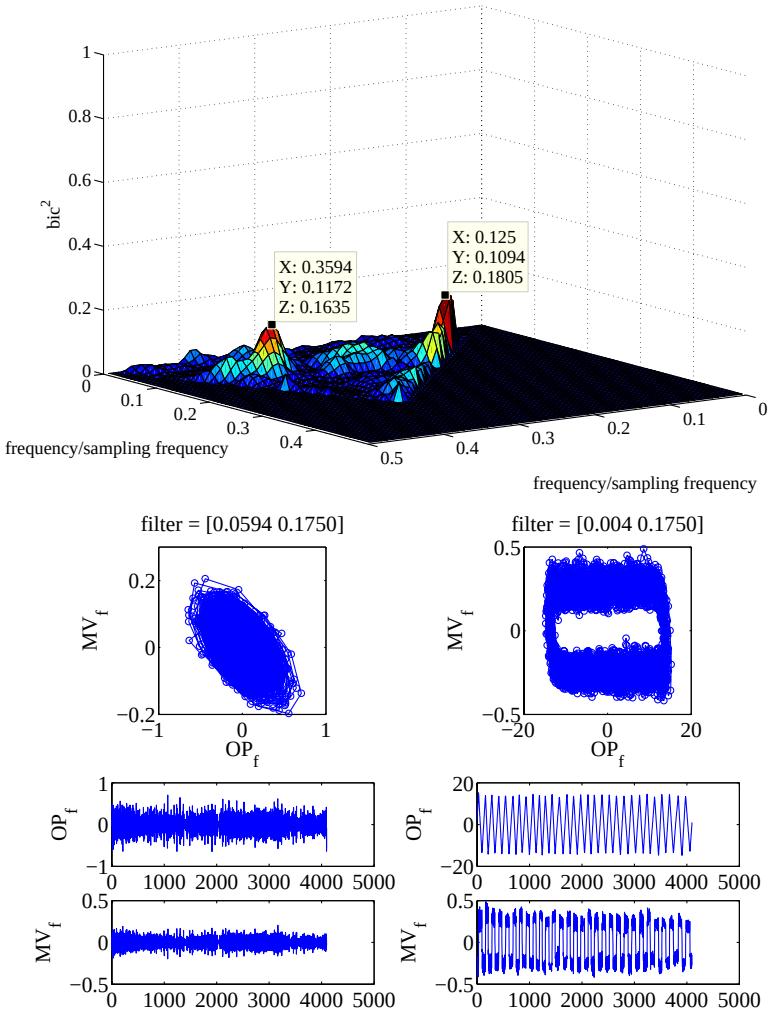

Figure 4.3: Upper figure shows bi-coherence plot for the loop, whereas the lower figure shows the filtered signals with different lower filter boundaries.

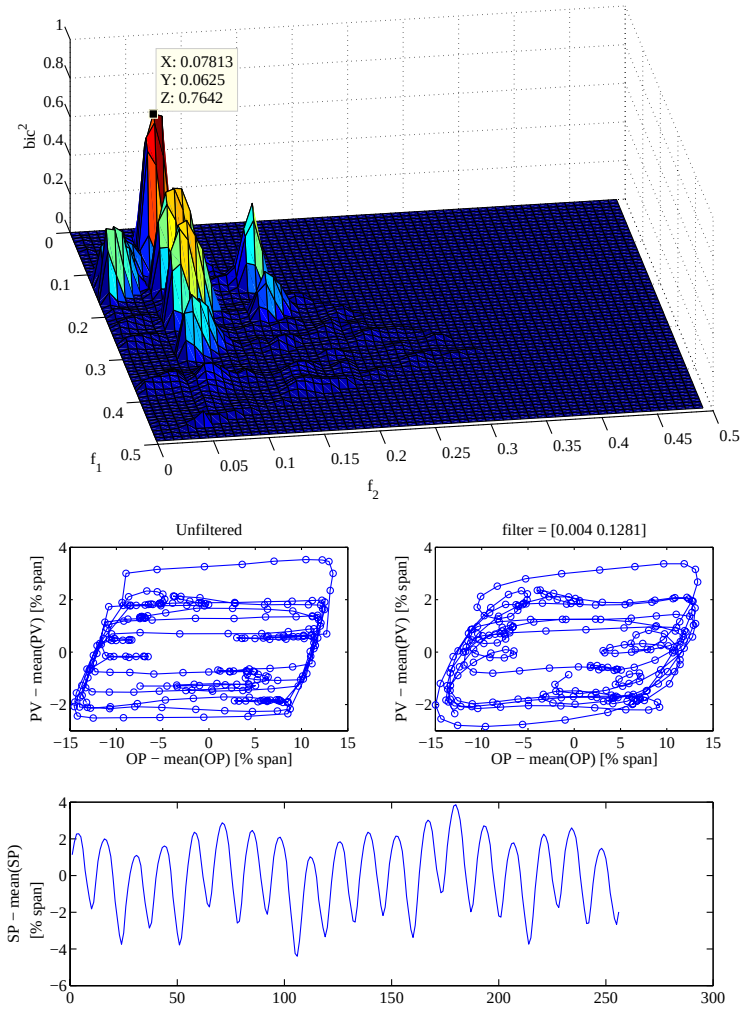

Figure 4.4: Upper figure shows the bi-coherence for the error signal $e(t)$ while the lower figure shows filtered and unfiltered patterns as well as the reference signal $r(t)$. For clarity only the first 256 point are plotted.

This initial investigation of the method lowered the hopes of implementing the method automatically, and research on other schemes is needed.

4.2 Fitting an ellipse to plant data

Even though the filtering proved to be difficult, I wanted to go on and investigate the fitting of an ellipse to the data. My first idea was to center the data and fit a parallelogram. However, this proved to be rather difficult and the idea was abandoned at an early stage.

This section will describe a simple algorithm for fitting an ellipse to a set of data (x_1, x_2) with length N to an ellipse. In [4] there was a method proposed for fitting. Some problems were encountered when implementing the method and therefore I include this chapter, which presents two more straightforward methods in addition to the one proposed in [4]. Matlab scripts/functions for all three methods can be found on the attached CD, see appendix D for info.

4.2.1 Simple unrotated ellipse

It can be shown[3] that the equation for a simple unrotated ellipse with center at the origin in the (x_1, x_2) plane is

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - 1 = 0. \tag{4.1}$$

The parameters a and b represent the half-axis in x_1 and x_2 directions respectively. Fitting this ellipse to the experimental data (x_1, x_2) means solving the equations

$$\begin{aligned} \theta_1 x_{1,1}^2 + \theta_2 x_{2,1}^2 &= 1 \\ \theta_1 x_{1,2}^2 + \theta_2 x_{2,2}^2 &= 1 \\ &\vdots \\ \theta_1 x_{1,N}^2 + \theta_2 x_{2,N}^2 &= 1. \end{aligned}$$

The least squares solution of this the system is the solution of $A^T A\theta = A^T b$, where $A = [x_1, x_2]$, $\theta^T = [\theta_1, \theta_2] = [1/a^2, 1/b^2]$ and $b^T = [1, 1, \ldots, 1]$. Geometrically speaking we find the vector $\theta = [\theta_1 \quad \theta_2]$ such that $A\theta$ is the projection of b into the column space of A.

[3] Use for instance `www.wikipedia.org` and search for "ellipse".

4.2.2 General conic

The references for the following are [29] and [4]. This method was proposed in [4], but some problems were encountered when I tried to apply it. This will be shown later, first a review of the ideas:

A general conic in the (x_1, x_2) coordinate system can be described by the equation

$$a_1 x_1^2 + a_{12} x_1 x_2 + a_2 x_2^2 + b_1 x_1 + b_2 x_2 + c = 0 \tag{4.2}$$

In [4] they suggest to find the least squares solution to this problem by minimizing $\|\Phi\theta\|$ subject to $\|\theta\| = 1$. Here Φ is the "observation matrix" $\Phi = [(x_1 \otimes x_1), (x_1 \otimes x_2), (x_2 \otimes x_2), x_1, x_2, e]$, where $\otimes$ in this case means "vector element-by-element" multiplication, and e is a vector of ones of appropriate size. θ is a vector with the parameters $[a_1, a_{12}, a_2, b_1, b_2, c]$. The proposed optimization is solved by optimizing on 6 parameters. Note that by dividing by c in equation (4.2) we reduce the degree of freedom to 5, and we don't need to solve the constrained least squares problem.

Another possible solution to avoid the constrained problem is simply to specify c. In analogy to the simple case of a centered unrotated ellipse it seems reasonable to set the constant $c = -1$, thereafter solving the system by unconstrained least squares, i.e. solving $\Phi^T \Phi \theta' = \Phi^T e$, where θ' is the 5 first elements of the original θ and $e = -c$ is a vector of ones of appropriate size. (This will be known as least squares solution of the general conic equation with $c = -1$.)

Interpretation of the results

Assuming that a solution of the problem stated above is found, one can interpret the results by using some general results from linear algebra. Equation (4.2) can be rewritten to

$$x^T A x + b^T x + c = 0$$

with $A = \begin{bmatrix} a_1 & a_{12}/2 \\ a_{12}/2 & a_2 \end{bmatrix}$, $b = [b_1\, b_2]$, $x = [x_1\, x_2]$, and still $c \in \mathscr{R}$. Let us use the following transformation for changing the coordinate system $(x_1, x_2) \in (X, Y)$ to $(\bar{x_1}, \bar{x_2}) \in (\bar{X}, \bar{Y})$:

$$x = Q\bar{x} + t,$$

where Q is a rotation matrix and t is a vector representing translation. Using the transformation gives the following equation in the new variable set:

$$\bar{x}^T \bar{A} \bar{x} + \bar{b}^T + \bar{c} = 0$$

where

$$\begin{aligned} \bar{A} &= Q^T A Q \\ \bar{b}^T &= \left(2t^T A + b^T\right) Q \\ \bar{c} &= t^T A t + b^T + c \end{aligned}$$

The rotation matrix Q can be chosen such that $\bar{A}$ is a diagonal matrix with the eigenvalues λ_1, and λ_2 of A on its diagonal. This can be achieved by choosing Q to be a vector of the corresponding eigenvectors of A.

If the conic is an ellipse with its center at the origin in $(\bar{X}, \bar{Y})$, then $\bar{b} = 0$ and hence the equation for the ellipse is $\bar{x}^T \bar{A} \bar{x} + \bar{c} = 0$, or $\lambda_1 \bar{x}_1^2 + \lambda_2 \bar{x}_2^2 + \bar{c} = 0$, or in analogy to the simple ellipse,

$$\frac{\bar{x}_1^2}{m^2} + \frac{\bar{x}_2^2}{n^2} = 1,$$

with

$$m = \sqrt{\frac{-\bar{c}}{\lambda_1}}, \qquad n = \sqrt{\frac{-\bar{c}}{\lambda_2}}.$$

Hence, to get an ellipse we need A to be positive definite and $\bar{c} < 0$.

4.2.3 Comparison of different fitting techniques

This section will consist of a qualitative comparison of the three methods on a chosen data set. (Loop B in figure 4.1.)

1. Fitting a simple unrotated ellipse to the centered data set (x_1, x_2) by finding the least squares approximation to equation (4.1).
2. Finding the least squares solution to the general conic equation (4.2) by setting $c = -1$.
3. Solving the constrained optimization problem in 6 variables subject to $\|\theta\| = 1$.

The results of the differnt algorithms are shown in figure 4.5.

Applying the three methods to the centered data yields rather similar results in for this data set, with the axes-lengths and orientation of the ellipsis being similar. For the case of uncentered data the situation is not so clear. (The simple ellipse is infeasible in this case.) The "direct solution" with specifying $c = -1$ and solving the unconstrained least squares yields a fit that resembles the data, though possibly with too large axes in the OP-variable. For the case of constrained optimization with $\|\theta\| = 1$ one observes that the fitted ellipse badly describes the

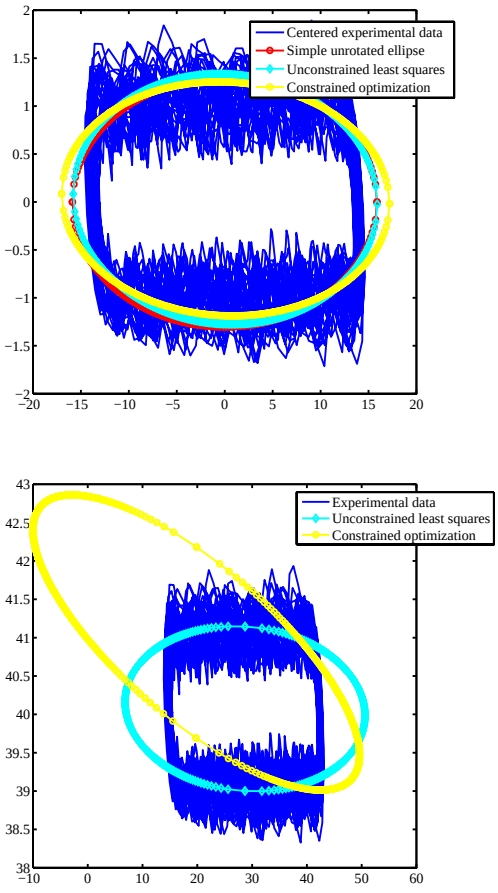

Figure 4.5: Comparison of the fitting techniques for centered data. The upper figure shows fitting to centered data, whereas the lower shows fitting for unentered data.

plant data. The vector norms were $\|\Phi\theta\|_2^{c=-1} = 0.0234$ and $\|\Phi\theta\|_2^{\|\theta\|=1} = 0.0512$. The constrained optimization had to "move away" from the solution with $c = -1$ (which was used as a starting point) because it did not exactly match $\|\theta\| = 1$. By investigation of the optimization it was evident that the problem was not convex in θ, as the routine (`fmincon` in MATLAB) moved increased the function value in many of the steps.

For other data set the optimization technique yielded infeasible solutions (imaginary axes), while the "direct solution" with $c = -1$ always gave an acceptable fit of the data (by inspection of the resulting data).

4.2.4 Discussion

The method for fitting of an ellipse to fitted data proposed in [4] might not always yield good results if it is implemented without modifications. Specifying the norm of the parameter vector to be $\|\theta\| = 1$ may not be the best choice for "closing" the problem. This choice does not guarantee that the fitted conic will be an ellipse. As seen from the equations for the unrotated centered ellipse in $(\bar{X}, \bar{Y})$, to have an ellipse one needs have A has eigenvalues grater than zero (i.e. positive definite) and $\bar{c} < 0$. The last constraint can, using the equations give earlier, be written as $c < \frac{1}{4} b^T A^{-1} b$. The reason why the constraint $\|\theta\| = 1$ fails in some cases may be that we are fitting an *ellipse* and not a *general conic*. All ellipses will lie in the subspace of the general conics, but using only $\|\theta\| = 1$ does not ensure that we get into this subspace. On the other hand, specifying $c = -1$ seems more promising, as we always got an ellipse in the cases studied.

4.2.5 Further work

Further work on this subject will be to derive the equations that ensure an ellipse and compare more closely with $\|\theta\| = 1$. Is there other specifications that will yield better fittings? The usual vector 2-norm was used in this report, but maybe there is better norms to use?

An easy way to bring this project forward is to contact the authors of [4] and ask them why they choose this approach,

Thereafter there is a need for a short routine that calculates the distance in OP for the rotated ellipse.

4.3 Overall discussion and conclusions

This chapter may be used as a starting-point for further work on solving the quantification problem. Still this problem is very open and due to time constraints no

more contributions can be given from my side. Good knowledge of mathematics is of course necessary for finding efficient solutions to these problems, justifying that the author (myself) should attend more math classes.

When the results are missing one can always focus of learned issues. Working with this problem I got insight into Wiener filtering, the bi-coherence function and general time-domain $\rightarrow$ frequency domain problems.

Chapter 5

Conclusions

The largest part of this masters thesis in Chemical Engineering was to use the Yam method on simulated and plant data. The method proved to be good, and the results were submitted as a conference paper at ANIPLA 2006, an international conference on automation (See appendix E). Thereafter some introductory work was conducted on quantification of stiction.

The first sections will consist of conclusions, then some recommendations to further work follow.

5.1 Yam method

The main message this work can give is that the Yam method is good to detect and report flow loops with obvious stiction. When posititoner data gets available the method should be applicable to all loops. Of 216 industrial loops investigated 167 of them were flow loops, a rather high number. The structure of the loops are not known, but assuming that some of them are cascaded loops and also data for the outer loops were available, the fraction of loops to be used with the Yam method is actually higher than $167/216$, since using any stiction detection technique on outer loops is in my opinion faulty use of the methods.

After application of the method to industrial data some new patterns were found that were not reported in literature as “expected patterns”. Chapter 3 tried to find a physical explanation for these patterns. Using basic control equations most of the patterns were shown to be “feasible” from a theoretical point of view. This may suggest that the Yam method can be extended to cover these new patterns, though more research and connection to experiments is needed.

5.2 Quantification

The work conducted on quantification consisted of trying to implement a method proposed by Choudhury et al. [4]. The method was found to be difficult to use on our available industrial data. Emphasis was put on filtering and fitting of an ellipse. Both issues had their problems. This means that the automatic quantification routine still is an open issue.

5.3 Recommendations to further work

The Yam method has been tested on both simulated and plant data. As more plant data gets available, is it desirable to continue this testing to make sure that these preliminary results for about 170 loops are valid.

Another issue which I find interesting is to investigate in more depth the mapping from the recored data to the symbols. For the Yam method one is trying to map the data into a vector space spanned by the vectors increasing (I), decreasing (D) or steady (S). What are the properties of this vector space, and of the mapping from $\mathscr{R}^N$ into this space? The next question is, what happens to the properties of the space if we extend the basis to the 7 primers used in [18]? (See page 13 for a figure of the primers.) Also it would be interesting to see if those 7 primers (vectors) represent an orthonormal basis for that vector space. A possible outcome of this work could be to identify more consistent sets of basis vectors and mappings, for instance by performing an orthogonalizatoin.

N is the length of observations.

For patterns in valves suffering from stiction, experimental work is needed. A cooperation with Enel is initiated focusing on diagnostication of valves and hopefully the results will be fruitful. If one is able to draw parallels to the "theoretical" chapter 3 one should think about extending the method to cover these new methods. The current work has shown that for the Yam method to be sharp and robust to noise a lot of patterns should be subtracted when calculating the (final) ρ_3 index. This should be kept in mind when extending the method; Introducing new patterns should be done with specific indices for those patterns to keep them as sharp as possible.

The quantification is a very open issue and a lot of work is needed. The work conducted and documented in this thesis is merely a small beginning. Some questions might be answered before staring more work:

- Is filtering of the data necessary? If yes, are there different approaches for filtering? What is the most suitable routine?
- Is it optimal to fit an ellipse to the data? What about a rectangle? If fitting an ellipse seems to be more easy and reliable, which technique should be

used to calculate the ellipsis-parameters fast and robust?

Bibliography

[1] Control valve dynamic specification (version 3.0). Technical report, EnTech Control Engineering Inc., 1998.

[2] A. Araújo and S. Skogestad. Controllability of processes with large gains. In *16th IFAC World Congress, Prague, 4-8 July*, 2005.

[3] B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7):1038–1138, 1994.

[4] M.A.A. Shoukat Choudhury, S.L. Shah, N.F. Thornhill, and David S. Shook. Automatic detection and quantification of stiction in control valves. Article in press.

[5] M.A.A Shoukat Choudhury, N.F. Thornhill, and S.L. Shah. Modelling valve stiction. *Control Engineering Practice*, 13:641–658, 2005.

[6] M.A.A.S. Choudhury, S.L. Shah, and N.F. Thornhill. Diagnosis of poor control-loop perfromance using higher-order statistics. *automatica*, 40: 1719–1728, 2004.

[7] Md. A. A. S. Choudhury. *Detection and Diagnosis of Control Loop Nonlinearities, Valve Stiction and Data Compression*. PhD thesis, Department of Chemical & Materials Engineering, Univeristy of Alberta, 2004.

[8] T. Hägglund. A friction compensator for pneumatic control valves. *Journal of Process Control*, 12:897–904, 2002.

[9] A. Horch. A simple method for detection of stiction in control valves. *Control Engineering Practice*, 7:1221–1231, 1999.

[10] M. Hovd. Compendium in course ttk18. NTNU, fall 2005.

[11] A. Ivaska. Estimation of the optimal sampling frequency for process analyzers. *Analytica Chimica Acta*, 190:89–97, 1986.

[12] M. Kano, H. Maruta, H. Kugemoto, and K. Shimizu. Practical model and detection algorithm for valve stiction. In *IFAC Symposium on Dynamics and Control of Process Systems (Dycops), CD-ROM, Cambridge*, 2004.

[13] D. Karnhopp. Computer simulation of stic-slip friction in mechanical dynamic systems. *ASME, Journal of Dynamic Systems, Measurement and Control*, 10:100–103, 1985.

[14] M. Karpenko, N. Sepehri, and D. Scuse. Diagnosis of process acutator faults using a multilayer neural network. *Control Engineering Practise*, 11:1289–1299, 2003.

[15] A. Kay and F. J. Doyle III. Friction compensation for a process control valve. *Control Engineering Practice*, 8:799–812, 2000.

[16] W. H. Press, S. A. Teukolsky, W. H. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran 77*, volume 1. Cambridge Univeristy Press, second edition, 1992.

[17] S. J. Qin and J. Yu. Multivariable controller perfomance monitoring. In *Preprints of: IFAC-ADCHEM-2006 Int. Conf.: "Advanced Control of Chemical Processes", Gramado (BR)*, volume 2, pages 593–600, 2006.

[18] R. Rengaswamy, T.Hägglund, and V. Venkatasubramanian. A qualitatice shape analysis formalism for monitoring control loop performance. *Engineering Applications of Aritficial Intelligence*, 14:23–33, 2001.

[19] R. Rengaswamy and V. Venkatasubramanian. A syntatic pattern-recognition approach for process monitoring and fault diagnostics. *Engineering Applications of Artificial Intelligence*, 8(1):35–51, 1995.

[20] M. Rossi and C. Scali. A comparison of techniques for automatic detection of stiction: simulation and application to industrial data. *Journal of Process Control*, 15:505–514, 2005.

[21] M. Rossi, C. Scali, and A. Farina. Implementazione di un sistema di monitoraggio e di individuazione di malfunzionamento degli attuatori.

[22] C. Scali and M. Rossi. An automatic system for closed loop performance monitoring. In *Proceedings of IcheaP-7, "7th Int. Conf. on Chemical and Process Engineering", in AIDIC Conference Series, ISBN 0390-2358*, volume 7, pages 321–330, 2005.

[23] C. Scali, F. Ulivari, and A. Farina. Issues in on-line implementation of a closed loop performance monitoring system. In *Preprints of: IFAC-ADCHEM-2006 Int. Conf.: "Advanced Control of Chemical Processes", Gramado (BR)*, volume 2, pages 687–692, 2006.

[24] S. Skogestad. Tuning for smooth pid control with acceptable disturbance rejection. Submitted to Ind.End.Chem.Res (2006).

[25] S. Skogestad. Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, 13:291–309, 2003.

[26] S. Skogestad and I. Postlethwaite. *Multivarible Feedback Control, Analysis and Design*. John Wiley and Sons, Ltd, 2nd edition, 2005.

[27] L. Smith. An introduction to neural networks. `http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html`.

[28] C. Stergiou and D. Siganos. Neural networks. `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.hmtl`.

[29] G. Strang. *Algebra lineare e sue applicazioni*. Liguori Editore, 1981.

[30] N. Thornhill, B. Huang, and H. Zhang. Detection of multiple oscillations in control loops. *Journal of Process Control*, 13:91–100, 2003.

[31] N. Tornhill and A. Horch. Advances in plant wide controller performance assessment. In *Preprints of: IFAC-ADCHEM-2006 Int. Conf.: "Advanced Control of Chemical Processes", Gramado (BR)*, volume 1, pages 29–36, 2006.

[32] H. Vedam and V. Venkatasubramanian. A wavelet theory - based adaptive trend analysis system for porcess monitoring and diagnosis. In *Proceedings of the American Control Conference*, pages 309–313, 1997.

[33] J. R. Whitely and J. F. Davis. Knowledge-based interpretation of sensor patterns. *Computers and chemical engineering*, 16(4):329–346, 1992.

[34] Y. Yamashita. An automatic method for detection of valve stiction in process control loops. *Control Engineering Practice*, 14:503–510, 2006.

[35] C. C. Yu. *Autotuning of PID Controllers*. Springer Verlag, 1999.

Appendix A

Yamashita stiction detection method in Matlab

A.1 Yamashita stiction detection method

Yamashita [34] has described a step-by-step procedure for implementing the stiction detection method. This appendix will show the corresponding MATLAB commands and functions necessary MATLAB functions. To get a better understand for the method I suggest to consult the publication by Yamashita [34]. The procedure is shown in table A.1. The files should be located at prof. Skogestads homepage. His current homepage is www.nt.ntnu.no/users/skoge/.

The following documentation was originally written to aid a convertion of the code to Microsoft's Visual Basic, but it also explains the general concepts of the code.

A.1.1 yamashita.m

The whole procedure may be implemented as shown under. Some comments are needed. First of all, step 0 and 9 is not necessary for an implementation the method in for example VISUAL BASIC (VB). Step 0 is a step for sorting out the user-set options and setting default values if no options were set. Note that the input argument `options` is a `struct`. If this is not possible in VB I assume there is another (maybe even more easy?) way, or one can simply skip this step. The report written out in step 9 is just the primes that were skipped as being regarded as "steady-steady" in both variables, and some other information. (Run the function to see). This is not strictly necessary for implementation, but could be useful in a debugging phase. For instance, if all symbols were set to "steady-steady", but we know that the loop is rather oscillatory and there is not much

Step	Procedure	MATLAB command
1	Obtain a time series of the controller output and valve position (or corresponding flowrates)	`x = OP; y = MV`
2	Calculate the time difference for each measurement variable	`dxdt = diff(x); dydt = diff(y)`
3	Normalize the difference values using the mean and standard deviation	`X = (dxdt - mean(dxdt))/std(dxdt); Y = (dydt - mean(dydt))/std(dydt)`
4	Quantize each variable in three symbols	`X_sym = primer(X), Y_sym = primer(Y)`
5	Describe qualitative movements in $x-y$ plots by combining symbolic values of each variable	`symbols_raw = collect(X_sym,Y_sym)`
6	Skip SS patterns from the symbol sequence	`symbols = skip(symbols_raw)`
7	Evaluate the index ρ_1 by counting IS and DS periods in the patterns found	`rho_1 = rho1(symbols)`
8	Find specific patterns and count stuck periods. Then evaluate the index ρ_3	`rho_3 = rho3(symbols)`

Table A.1: Yamashita stiction detection method with corresponding MATLAB commands.

noise, something is wrong with the implementation. For the reader who has not worked with the Yamashita stiction detection method this may seem a bit cryptic, but I have found that printing these numbers may be useful when testing out the method.

The original publication by Yamashita [34] does not consider if the valve has `direct` or `reverse` action. However, during my work I have found that a Yamashita modification should be used for valves with reverse action. For these valves the index ρ_3 should be calculated on the basis of (OP,−MV) rather than (OP,MV). This is the reason for the option `direct` or `reverse` action valve in the function. See chapter 3 for details.

```
function results = yamashita(OP,MV,options)
% function results = yamashita(OP,MV,options)
%
% This file is a simple implementation
% of the Yamashita stiction detection metod
% in Matlab
%
% need: [diff.m], [std.m], [mean.m],
%       primer.m, collect.m,
%       skip.m, rho1.m, rho3.m
%
%       [fun.m] means fun.m is an internal
%       matlab function
%
% options: options.direct = 1  : direct action  (default)
%                         = -1 : reverse action
%          options.report = 1  : make a small report
%                         = 0  : no report      (default)
%
%
%                   Henrik Manum, 17 march 2006

% step 0. Check if there are some options set by user
if nargin == 2
    options.report = 0;
    options.direct = 1;
else
    if isfield(options,'report') == 0
        options.report = 0;
    end
    if isfield(options,'direct') == 0
        options.direct = 1;
```

```
    end
end


% The method strictly follows the steps of Yamashita
% step 1. Obtain a time series of the controller
% output and valve position (given by input to file)
x = OP; y = options.direct*MV;

% step 2. Calculate the time difference for each
% measurement variable
dxdt = diff(x); dydt = diff(y);

% step 3. Normalize the difference vaules using
% the mean and standard deviation
X = (dxdt - mean(dxdt))/std(dxdt);
Y = (dydt - mean(dydt))/std(dydt);

% step 4. Quantize each variable in three symbols
X_sym = primer(X);
Y_sym = primer(Y);

% step 5. Describe qualitative movements in x-y
% plots by combing symbolic values of each variable
symbols_raw = collect(X_sym,Y_sym);

% step 6. Skip SS patterns from the symbol sequence
symbols = skip(symbols_raw);

% step 7. Evaluate the index rho_1 by counting IS and
% DS periods in the pattern found
results.rho_1 = rho1(symbols);

% step 8. Find specific patterns and count stuck
% periods. Then evaulate the index rho_3
results.rho_3 = rho3(symbols,options.direct);

% step 9. If wanted, make a small report
if options.report
    results.report = check_results(symbols,symbols_raw);
end

return
```

Step 1-3 uses normal MATLAB commands. Steps 4-8 uses the following functions:

A.1.2 primer.m

This is the core of the routine as it translates the derivatives of the input signals to symbolic values.

```
function vect = primer(x)
% function symbols_out = primer(x)
%
% The function finds a symbolic representation
% of the vector x for use in Yamashita
% stiction detection method. The standard deviation
% is used as a thershold.
%
%       Henrik Manum 10.feb.2006
x_std = std(x);   % threshold (should be about 1 if the
                  %            variable is N(0,1))
for i = 1:length(x)
    if x(i) > x_std
        vect(i) = 'I';
    elseif x(i) < (-1)*x_std
        vect(i) = 'D';
    else
        vect(i) = 'S';
    end
end

return
```

A.1.3 collect.m

This is a function to collect the symbolic values into a symbolic array. In MATLAB I choose to use a `cell-array` here.

```
function output = collect(x,y)
% function output = collect(x,y)
% make an array of symbols according to table 1
% in the publication by Yamashita
for i = 1:length(x)
    output{i} = [x(i) y(i)];
```

```
end

return
```

A.1.4 skip.m

This function removes the 'SS' symbols from the series of symbols

```
function symbols = skip(symbols_raw)
% Function that removes 'SS' symbols from the symbolic
% represenatation of the time series
%               22. feb 2006, Henrik Manum
n = 1; symbols{1} = [];
for i = 1:length(symbols_raw)
    if (symbols_raw{i}(1) == 'S')&(symbols_raw{i}(2) == 'S')
    else
        symbols{n} = symbols_raw{i};
        n = n + 1;
    end
end
return
```

A.1.5 rho1.m

Calculation of ρ_1 as defined by Yamashita [34].

```
function rho_1 = rho1(x)
% function rho_1 = rho1(x)
% calculates the rho_1 index defined by yamashita for
% stiction detection.

% count the 'IS' and 'DS' periods found
n = 0;   % counter
for i = 1:length(x)
    if (x{i}(1) == 'I')&(x{i}(2) == 'S')
        n = n + 1;
    end
    if (x{i}(1) == 'D')&(x{i}(2) == 'S')
        n = n + 1;
    end
end
rho_1 = n/length(x);

return
```

A.1.6 rho3.m

Calculation of ρ_3 as defined by Yamashita [34].

```
function rho_3 = rho3(x,direct)
% function rho_3 = rho3(x)
% Evaluation of the index rho3 defined by yamashita

m = 0; m1 = 0; m2 = 0;    % counters
DS = 0; IS = 0;           % tags
for i = 1:(length(x) - 1)
    if (x{i}(1) == 'I')&(x{i}(2) == 'S')
        if IS == 1      % the last was also IS
            m1 = m1 + 1;
        else
            IS = 1;     % switch the IS tag on
            DS = 0;     % switch the DS tag off
            m1 = 1;     % start counter
            m2 = 0;     % reset the other counter
        end
        if (x{i+1}(1) == 'D')&(x{i+1}(2) == 'D')|...
                (x{i+1}(1) == 'D')&(x{i+1}(2) == 'I')|...
                (x{i+1}(1) == 'S')&(x{i+1}(2) == 'D')|...
                (x{i+1}(1) == 'I')&(x{i+1}(2) == 'D')|...
                (x{i+1}(1) == 'D')&(x{i+1}(2) == 'S')
            m = m + m1;
        end
    elseif (x{i}(1) == 'D')&(x{i}(2) == 'S')
        if DS == 1      % the last was also DS
            m2 = m2 + 1;
        else
            DS = 1;     % switch the DS tag on
            IS = 0;     % switch the IS tag off
            m2 = 1;     % start counter
            m1 = 0;     % reset the other counter
        end
        if (x{i+1}(1) == 'D')&(x{i+1}(2) == 'I')|...
                (x{i+1}(1) == 'S')&(x{i+1}(2) == 'I')|...
                (x{i+1}(1) == 'I')&(x{i+1}(2) == 'D')|...
                (x{i+1}(1) == 'I')&(x{i+1}(2) == 'I')|...
                (x{i+1}(1) == 'I')&(x{i+1}(2) == 'S')
            m = m + m2;
        end
    else
```

```
        IS = 0;     % neither IS
        DS = 0;     % nor DS
        m1 = 0;     % reset counters
        m2 = 0;
    end
end
rho_3 = rho1(x) - m/length(x);
return
```

A.1.7 check_results.m

This function is not necessary for using the Yamashita method, but it may be handy when performing simulations as it runs a Yamashita check on the results to see if either the valve or the command signal was steady all the time. This is the function which generates the small report mentioned above.

```
function output = check_results(x,y)
% a simple signal check-up. This function
% is optional, maybe good for debugging
% purposes, but could be left out in the
% final implementation
n = length(x);
m1 = 0;
m2 = 0;
for i = 1:length(x)
    if x{i}(1) == 'S'
        m1 = m1 + 1;
    end
    if x{i}(2) == 'S'
        m2 = m2 + 1;
    end
end

if m1 == n
    output.OP = 'ALL_S';
    %error('OP all S');
else
    output.OP = 'OK';
end
if m2 == n
    output.MV = 'ALL_S';
    %error('MV all S');
else
```

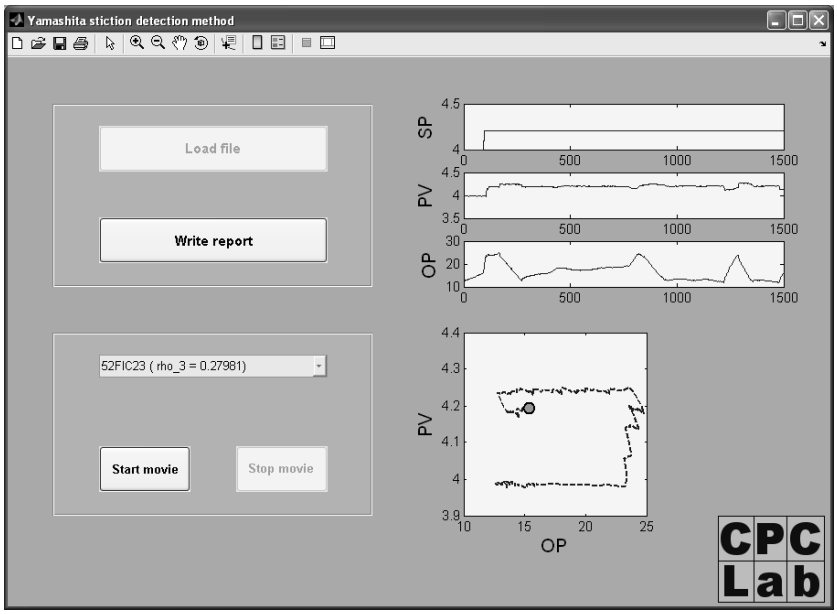

Figure A.1: Snap-shot of the GUI.

```
    output.MV = 'OK';
end
output.length_raw = length(y);
output.length = n;
output.length_S_OP = m1;
output.length_S_MV = m2;
return
```

A.2 GUI

In addition to using the command-line application in Matlab there was a need for a graphical user interface (GUI). The GUI is rather simple, with the following features

- Read the data from an Excel file with data saved in the same format as the PCU (the other, more comprehensive, process control monitoring system).
  - When the data is loaded the ρ_3 index is calculated for all the loops.
  - The upper limit for number of loops is not known, but 10-15 loops are no problem to be analyzed simultaneously.
- Plot the recorded data.

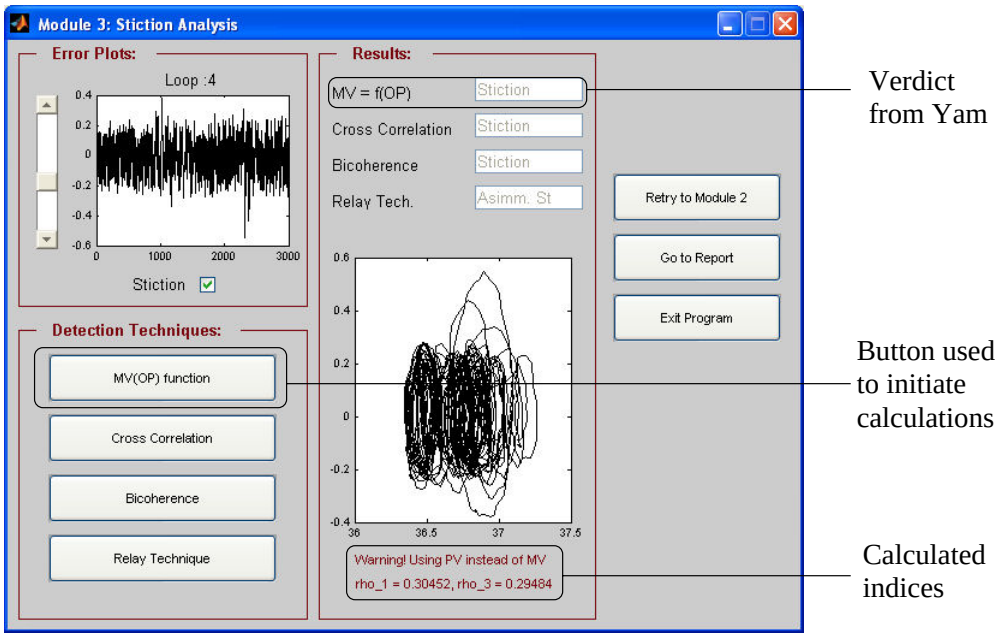

Figure A.2: PCU's stiction module with Yam method implemented under button "MV(OP) function".

- Display the MV(OP) plot as a function of time.
- Write a report with loop names, ρ_3's and the word "stiction" if $\rho_3 > 0.25$.

Figure A.1 shows a screen-shot of the GUI. There should be no need for further presentation. The code is included on the CD and is written with the aid of the Matlab documentation. Matlab's "GUIDE" was not used, I found it easier to simply write the code myself.

A.3 PCU with Yam

In addition to the stand-alone GUI described in the previous section, I added the Yam routines to the existing PCU as an example of how the method can be implemented. As this software is still under development it is not included on the CD. Figure A.2 shows the main changes seen from PCU's GUI. As seen, the user now has an extra method for stiction evaulation. If $\rho_3 > 0.25$, stiction is reported. If not, "no indication" is shown. The indices ρ_1 and ρ_3 are shown in the lower part of the GUI.

Appendix B

Normalization invariant to scaling

Want to show that when normalizing the differentials by subtracting the mean and dividing by the standard deviation it is not important if the recored signals were normalized to be between 0 and 1 or 0 and 100 by dividing by the ranges.

Let x be a recored signal, $x = (x_j)_{j=1}^N$, where N is the number of elements in x. Then the differentials of x can be defined as $\dot{x} = (x_j - x_{j-1})_{j=2}^N$. Further the scaled signal is $\tilde{x} = (1/\Delta x)x$.

By inspecting the Matlab documentation, the formulas "mean" and "std" does the following operations:

$$\text{mean}(x) = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{B.1}$$

$$\text{std}(x) = \sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{B.2}$$

Hence, for the scaled signal $(1/(\Delta x))x$ the normalization is

$$\begin{aligned}
\tilde{\dot{x}}^N &= \frac{\frac{1}{\Delta x}\dot{x} - \frac{1}{N}\frac{1}{\Delta x}\sum_{i=1}^{N}\dot{x}_i}{\sqrt{\frac{1}{N-1}\sum_{j=1}^{N}\left(\frac{1}{\Delta x}\dot{x}_j - \frac{1}{N}\frac{1}{\Delta x}\sum_{k=1}^{N}\dot{x}_k\right)^2}} = \\
&= \frac{\frac{1}{\Delta x}\dot{x} - \frac{1}{N}\frac{1}{\Delta x}\sum_{i=1}^{N}\dot{x}_i}{\sqrt{\left(\frac{1}{\Delta x}\right)^2\frac{1}{N}\sum_{j=1}^{N}\left(\dot{x}_j - \frac{1}{N}\sum_{k=1}^{N}\dot{x}_k\right)^2}} = \\
&= \frac{\dot{x} - \frac{1}{N}\sum_{i=1}^{N}\dot{x}_i}{\sqrt{\frac{1}{N-1}\sum_{j=1}^{N}\left(\dot{x}_j - \frac{1}{N}\sum_{k=1}^{N}\dot{x}_k\right)^2}} = \\
&= \frac{\dot{x} - \bar{\dot{x}}}{\sigma_{\dot{x}}} = \\
&= \dot{x}^N,
\end{aligned}$$

so the normalization is invariant to scaling.

About the notation used over, $x \in \mathscr{R}^N$, while $\bar{x}$ and σ_x are scalars. However, this notation is in agreement with the commands entered in Matlab, were Matlab multiplies with unit vectors when necessary.

Appendix C

Wiener filtering using the FFT

Want to use the internal Matlab function "fft" (Fast Fourier Transform) to calculate an approximate Wiener filter. The idea is fairly simple:

1. Take the FFT of the recorded signal.
2. Set all unwanted coefficients to zero.
3. Take the inverse FFT to get the filtered signal.

In the Matlab documentation one finds that "The functions `X = fft(x)`" and `x = ifft(X)` implement the transform and inverse transform pair given for vectors of length N by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)} \tag{C.1}$$

$$x(j) = \frac{1}{N}\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)} \tag{C.2}$$

where

$$\omega_N = e^{-2\pi i}/N$$

is an N^{th} root of unity."

In practice, given a vector x, which ideally is a power of two, but here will be assumed to be even, sampled in time-domain with sampling interval $\Delta = T_s$, $X = \text{fft}(x)$ will have the following structure:

$$X = [c(\omega = 0) \quad c(\omega = \frac{1}{\Delta N}\cdot 1) \quad c(\omega = \frac{1}{\Delta N}\cdot 2) \quad \dots \quad c(\omega = \frac{1}{\Delta N}\cdot(\frac{N}{2}-1)) \quad \dots$$
$$\dots \quad c(\omega = \frac{1}{2\Delta}) \quad c(\omega = -\frac{1}{\Delta N}\cdot(\frac{N}{2}-1)) \quad c(\omega = -\frac{1}{\Delta N}\cdot(\frac{N}{2}-2)) \quad \dots \quad c(\omega = -\frac{1}{\Delta N}\cdot 1)]$$

This is the so-called "wrap-around" order of the frequencies. One sees that the coefficients increase linearly from the DC (direct current/steady state) to the Nyquist frequency $\frac{1}{2\Delta}$, then the negative frequencies follow in a decreasing order.

Assuming now there is an interval (ω_L, ω_H) for which we want to keep the coefficients, setting all coefficients outside equal to zero. For the positive frequencies we have that

$$\omega(k) = \frac{1}{\Delta N}(k-1), \qquad k = 2,3,\ldots,\frac{N}{2}-1.$$

So, the limits in terms of vector position k will be

$$k_L = \Delta N \omega_L + 1$$
$$k_H = \Delta N \omega_H + 1$$

With the current knowledge the following Matlab script, an implementation of the filter, should be easy to follow:

```
function y = wfilter(x,f)
% Simple function that takes in the signal x
% which has an even length,
% and filter boundaries f = [f1 f2]
% and returns the filtered signal y.
% The algorithm is to take the DFT of x,
% setting all cofficients outside the limits
% to zero and return the inverse-transformed y
%
%                                  Henrik Manum, may 2006 (Pisa)


N = length(x);

y = fft(x,N);
k = [floor(f(1)*N + 1) ceil(f(2)*N+1)];
y(1:k(1)-1) = 0 ;
y(k(2)+1:(N+2-k(2)-1)) = 0;
y(N+2-k(1)+1:end)=0;
p = abs(y)/(N/2);
p = p(1:N/2).^2;
% freq = [0:1/(N-1):1];
freq = [0:N/2 - 1]/N;
plot(freq,p);

y = ifft(y);
```

```
return
```

Appendix D

Description of software

Relevant software for this project should be avaliable under prof. Skogestad's homepage. Currently the adress is www.nt.ntnu.no/users/skoge. Thereafter click on "Diploma Students", "2006" and finally "manum". In Pisa there should also be a CD avaliable. The contents are:

- Report in .pdf format
- A presenation of the thesis in .ppt format.
- Matlab scripts for performing the filtering and fitting discussed in chapter 4.
- Yam method implemented in Matlab and GUI.

In the Yamashita/GUI directory there are examples of how the Excel file should be. For more information about this format contact either me or prof. Scali.

Appendix E

Paper submitted to ANIPLA 2006

Paper submitted with title “Automatic Diagnosis of Valve Stiction by means of a Technique based on Shape Analysis Formalism”

Authors: Henrik Manum (NTNU) and Claudio Scali (Univeristy of Pisa).

Status: Submitted.

Abstract

Valve stiction is an important cause of performance deterioration in control loops of industrial plants; owing to the large number of loops in complex plants and different cause of poor performance, it is important to be able to detect causes and suggest actions to perform in automatic way. The paper examines some recent techniques to detect the presence of valve stiction as root causes of oscillations, by using qualitative shape analysis formalism. Basic properties and main factors are put into evidence by application on data generated by simulation, while the reliability is checked by application on plant data sets to account for sensitivity to noise, for the effect of set point variations due to cascade or advanced control acting on the upper level. The algorithm shows to be able to detect stiction when it shows up with clear patterns (about 50% of examined cases coming from about 200 data sets). This allows a quicker detection and saving of computation time with respect to more comprehensive techniques, thus suggesting on line implementation of the technique.

Keywords: Process control, Stiction detection, Pattern recognition, Shape analysis

E.1 Introduction

In the last years Closed Loop Performance Monitoring (CLPM) has attracted large interest in academic research and in industrial applications, as the possibility of detecting the onset of anomalies and determining causes of performance deterioration in base control loops is certainly of vital importance for the success of advanced control layers (Multivariable, Optimization). The goal is to develop fully automatic monitoring systems, able to analyze the large number of data coming from control loops (hundreds in an industrial process units) and to determine the cause of poor performance, thus indicating to the operator counteractions to perform on the plant. Performance deterioration can be due to different factors, ranging from incorrect design or tuning of controllers, to anomalies and failures of sensors, presence of friction in actuators, external perturbations, deteriorations in the process itself. It is evident that actions to perform on the plant are different depending on the cause, hence the importance of being able to distinguish them. Very often anomalies appear as oscillations in the process variable and the challenge is to trace back the origin: provenience (which loop?) and root (which cause?). The definition of reliable indexes and their applicability for the case of multivariable processes is certainly an open issue (see [31] and [17] for an updated review).

Actuators (valves in the large majority of control loops of industrial processes) are very often the most common reason of generation of oscillations, as the presence of friction distorts the relationship between the input (controller action) and the output (manipulated variable) from linear to non linear, thus originating limit cycles in the loop. This cause of performance deterioration is certainly more frequent than controller tuning, which is by far the most common issue addressed. While many different approaches and procedures for identification and controller retuning have been proposed and commercialized in the last years (see [35]), only very few techniques for the detection of stiction (as it is called among experts) are operating industrially, many basic issues still remain unresolved and are object of fervent research.

This paper is devoted to the diagnosis of stiction from plant data illustrating some new techniques and putting into evidence issues in the application on industrial data. Next sections will deal with: recalling basic issues of valve stiction and related models (section II), illustrating techniques for automatic recognition of stiction and in particular a newer one based on qualitative formalism analysis [34] (III), showing basic characteristics by application on simulated data (IV), checking its reliability on industrial data (V) and drawing some conclusions and indicating next work (VI).

E.2 VALVE STICTION

Choudhury et al. [5] conduct a review of past and present definitions of stiction. The review reveals a lack of a formal and general definition of stiction and the mechanism(s) that causes it. They therefore propose a new definition of stiction which will be used in this paper.

Fig. E.1 shows the movements of a typical sticky valve in a feedback loop with the valve output (manipulated variable (MV)) as a function of valve input (controller output (OP)). The movement consists of four components: deadband, stickband, slip-jump and the moving phase. When the valve comes to rest or changes direction at point A the valve sticks. After the control signal overcomes the deadband (AB) and the stickband (BC) the valve position jumps to point D. This jump is called a slip-jump. The valve may now move smoothly to point E, or it may stick again due to low or zero velocity while traveling in the same direction. In such a case, the magnitude of the deadband is zero and only stickband is present. The deadband and stickband represents the behavior of the valve when it is not moving, though the input signal is varying. Slip-jump represents the abrupt release of static energy when the static friction is overcome. Once the valve slips, it will move until it sticks again, under presence of dynamic friction that is a lot lower than the static friction [5].

On the basis of the four components defined above one can define stiction in the following way: *Stiction is a property of an element such that its smooth movement in response to a varying input is preceded by a sudden abrupt jump called the slip-jump. Slip-jump is expressed as a percentage of the output span. Its origin in a mechanical system is static friction which exceeds the friction during smooth movement* [5].

Choudhury et al. [5] shows how one can develop a physical model of stiction based on first principles. A complete review of models is reported in [3], certainly one of the most used is the Karnopp model [13]. A common factor for these models is that they require detailed knowledge about each valve, since factors such as spring constant and diaphragm area vary from valve to valve; in addition, other parameters (static and dynamic friction factors, contact area etc.) are unknown and subject to changes.

Due to the practical difficulties with using the first-principles-based model, [5] developed a model that depends only on the parameters S (stickband) and J (slip-jump), as defined graphically in figure E.1. This data-driven model will be used for simulations in this paper.

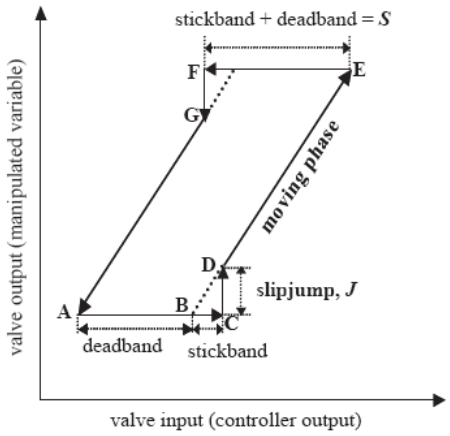

Figure E.1: Typical stiction pattern with definition of parameters S and J used in stiction model. Taken from [5].

E.3 AUTOMATIC DETECTION OF STICTION

E.3.1 Techniques based on PV(OP) - brief review

A number of automatic stiction detection methods have been proposed in the literature. A brief review of three popular methods that use PV-OP data as a basis for stiction detection follows.

The classic cross-correlation technique by Horch [9] is popular due to its simple implementation [20]. It is based on the cross-correlation between control input and process output. Given a control loop that is oscillating, the Horch method should be able to distinguish the two important causes "external oscillating disturbance" and "static friction (stiction)," because for external oscillating disturbance the phase lag in the cross-correlation is $-\pi$, while it is $-\pi/2$ for stiction.

Choudhury et al. [4] have proposed a method based on High Order Statistics. It is observed that the first and second order statics (mean, variance, autocorrelation, power spectrum etc.) are only sufficient to describe linear systems. Non-linear behavior must be detected using higher order statistic such as "bi-spectrum" and "bi-coherence." A stiction index is defined by means of how much non-linear behavior is present.

A technique based on fitting the recorded oscillations with three different signals: the output response of a first order plus time delay system under relay control, a triangular wave, and a sine wave is proposed in [20]. After evaluation of an error-norm between the fitted data and the recorded signal, a phenomenon is identified. Relay-control and triangular waves are associated with the presence

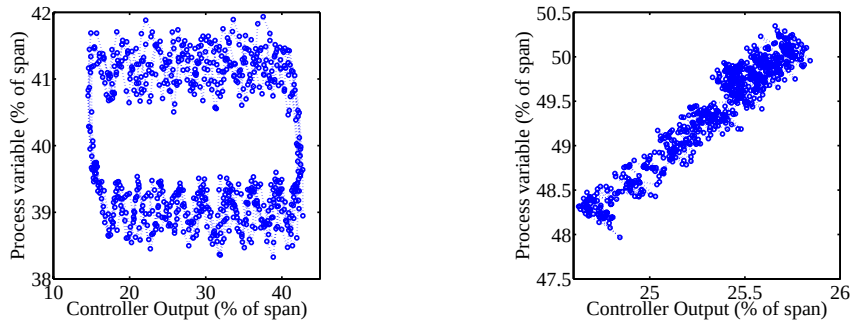

Figure E.2: Stiction pattern (upper figure) and good loop (lower figure) from plant data. In both figures 720 samples are plotted.

of stiction, whereas sine waves with external perturbations. The error-norm gives rise to a stiction index.

Rossi and Scali [20] performed a comparison of the techniques presented above. A major finding was that, according to process and stiction parameters, every technique has an uncertainty region where no decision can be taken in the absence of further information about the process. A sequential application of the three techniques is then suggested, starting from the cross-correlation (shortest computation time), to the relay based method (longer time).

E.3.2 Techniques based on qualitative description formalism

Fig. E.1 shows a typical stiction pattern as seen by plotting the valve position as a function of the valve input. For flow-loops, of which there are a vast number in the chemical process industries, one may assume that the flowrate is proportional to the valve position. This assumption will be used throughout this paper. Thus one expects to see a pattern as showed in fig. E.1 for flow loops with sticky valves in a (OP, PV)-plot. Two loops from industrial data are shown in fig. E.2, one with and one without the presence of stiction. By using the eyes humans can easily detect stiction in these loops, because of our excellent pattern-detection abilities. To to able to detect stiction automatically in a plot such as in fig. E.1 is the main idea behind the techniques based on qualitative description formalism.

A common feature of methods found in literature is that they try to describe the recorded signals using a set of fundamental units, called primitives. A recorded signal is described using the primitives, and after the primitives are found some procedure is used to interpret/compare the primitives with known phenomena, such as stiction.

An automated qualitative shape analysis (QSA) formalism for detection and

diagnosing different kinds of oscillations is presented in [18]. They use 7 primitives and a neural network to identify the primitives. The neural network is presented in [19].

A more detailed description of how to develop a neural network for use in pattern recognition in chemical process industries is shown in [33].

A strong argument for using a neural network in the identification of the primers is that usually recorded data is too noisy to be represented by symbols using simpler schemes. Yamashita [34] suggests a simpler identification scheme based on calculation of the time-differentials of a signal. The main motivation for the current work is to investigate if this method is applicable to industrial data. Due to its transparency and ease of programming this method is more preferable than the complicated methods using neural networks. Since this method is new, a detailed description follows.

E.3.3 The Yamashita stiction detection technique

The Yamashita stiction detection technique (Yam) [34] consists of a simple identification scheme using the differentials of the recorded signals and a representation of the signals using 3 primers. The identified series of primers are combined to form a time-series of movements which is the basis for calculation of a stiction index.

For a given time series signal the simplest way to describe the signal by symbols is to use the following three primers: increasing (I), decreasing (D), and steady (S). The primers can be identified using standard deviation of the differentials of the recorded signals as a threshold for identification. In words, the identification works like this:

1. Calculate the differentials of the given signals.
2. Normalize the differentials with the mean and standard deviation.
3. Quantize each variable in three symbols using the following scheme (x is the recored signal and $\dot{x}$ is the normalized differentials):
   - If $\dot{x} > 1$, x is increasing (I)
   - If $\dot{x} < -1$, x is decreasing (D)
   - If $-1 \leq \dot{x} \leq 1$, x is steady (S)

By combining the symbols for the OP and MV signals we get a symbolic representation of the development in an (OP, MV) plot with time. The primers for the combined plot are shown in table E.1. The sticky motions, $\boxed{\text{IS}}$ and $\boxed{\text{DS}}$ are framed. These are the two primers when the controller is either increasing (I) or

Table E.1: Primers for an (OP, MV) plot.

OP/MV	D	S	I
I	ID	IS	II
S	SD	SS	SI
D	DD	DS	DI

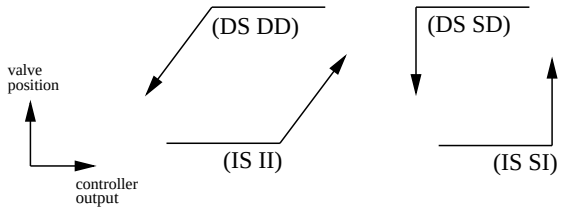

Figure E.3: Qualitative shapes found in sticky valves. Adopted from [34].

decreasing (D) its output, while the valve position is steady (S). Based on this a stiction index ρ_1 can be defined:

$$\rho_1 = (\tau_{\text{IS}} + \tau_{\text{DS}}) / (\tau_{\text{total}} - \tau_{\text{SS}}). \tag{E.1}$$

In (E.1) τ_{IS} is the total number of occurrences of the combined primer "IS", and so on. Note that the time when both the OP and MV are steady at the same time is removed. The sticky movement corresponds to 2 of 8 primers in table E.1, hence for a random signal $\rho_1 \approx 0.25$. If $\rho_1 > 0.25$ there could be stiction in the valve.

In applications ρ_1 is found to be not accurate enough to identify stiction, often it is high even though there is not stiction in the loop. This calls for an improved index. Fig. E.3 shows the typical qualitative shapes found in sticky valves in analogy with fig. E.1. Based on this, a refined index ρ_2 is defined as

$$\rho_2 = (\tau_{\text{IS II}} + \tau_{\text{IS SI}} + \tau_{\text{DS DD}} + \tau_{\text{DS SD}}) / (\tau_{\text{total}} - \tau_{\text{SS}}). \tag{E.2}$$

In (E.2) $\tau_{\text{IS II}}$ is the total number of IS samples in all the found (IS II) movements in the observation window, $\tau_{\text{DS DD}}$ is the total number of DS samples in the found (DS DD) movements, and so on. For example, if we had a time series (5·IS, 3·II), this counts as 5. We have that $\rho_2 \leq \rho_1$, where the equality holds when all the sticky motions in the valve corresponds to the shapes shown in figure E.3.

In the extreme case when the valve does not move, only patterns IS, SS and DS will be found. This special case will make $\rho_2 = 0$, not 1. To avoid this a new index ρ_3 is used, that is calculated by subtracting all the sticky patterns that does

not match the patterns shown in fig. E.3 from ρ_1:

$$\rho_3 = \rho_1 - \left(\sum_{x \in W} \tau_x \right) / \left(\tau_{\text{total}} - \tau_{\text{SS}} \right). \tag{E.3}$$

The set W contains all the patterns that have nothing to do with stiction. In symbols, these are W = {IS DD, IS DI, IS SD, IS DS, DS DI, DS SI, DS ID, DS II, DS IS}. For example, if the movement was (IS IS IS DD IS IS II), we should subtract $3/7$ from the original ρ_1, because the 3 first IS primers could not be a part of a stiction pattern since they were followed by a DD. Except for the special case when the valve does not move, $\rho_3 = \rho_2$.

Let us now summarize method and implementation [34]:

1. Obtain a time series of the controller output and valve position (or corresponding flowrate).
2. Calculate the time difference for each measurement variable.
3. Normalize the difference values using the mean and standard deviation.
4. Quantize each variable in three symbols.
5. Describe qualitative movements in (x,y) plots by combining symbolic values of each variable.
6. Skip SS patterns for the symbolic sequence.
7. Evaluate the index ρ_1 by counting IS and DS periods in the patterns found.
8. Find specific patterns and count stuck periods. Then evaluate the index ρ_3.

The method can easily be implemented in any suitable programming language.

E.4 APPLICATION ON SIMULATED DATA

Before testing the method on plant data it was desirable to use it on simulations to understand how it performs in various cases. For a possible industrial implementation it is important to get an understanding for how noise, external disturbances and set point changes affect the performance of the method.

In the simulated data we have the MV data available, and in this section the MV data were used as a basis for calculation of the indices, rather than the PV. From fig. E.4 one observes that PV is MV filtered by the process.

Before using the Yam method, three degrees of freedom needs to be specified:

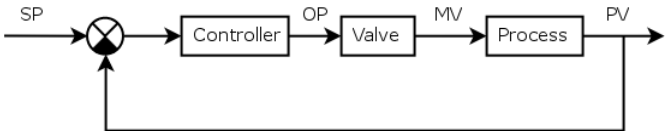

Figure E.4: Simple feedback scheme with definition of variables.

- Length of time window
- Threshold in symbolic representation
- Sampling time

Except for practical problems there is no upper limit for the time window. In this section there were always at least 3 cycles of oscillations. For the threshold the standard deviation of the time differentials was used, as recommended by Yamashita. The effects of altering the sampling time will be investigated in this section by altering the sampling time in the presence of noise in the recorded data.

Yamashita [34] writes that some prefilter can be used if the raw signals are very noisy. This was not used here, as keeping the method as clear and simple as possible was one of the motivations for applying the Yam method.

For the process, a first order plus delay model $g(s) = ke^{-\theta s}/(\tau s+1)$ was used. In the present simulations the slip-jump parameter $J = 1$, and the process time-constant $\tau = 10$ for all cases studied. For the controller a SIMC-tuned PI controller [26] was used, with control equation $c(s) = K_c(1+1/(\tau_1 s))$ and parameters $K_c = (1/k)(\tau/(\tau_c+\theta))$, $\tau_I = \min\{\tau, 4(\tau_c+\theta)\}$ and finally $\tau_c = \theta$. Here θ is the process delay, k is the process gain, K_c the controller gain, τ_I the integral time, and τ_c a tuning parameter with its recommended setting, equal to the effective delay [26].

E.4.1 Noise-free data

As a first attempt the method was applied on noise-free data generated by the Choudhury model. Rossi and Scali [20] computed stiction indices for the cross-correlation technique [9], bi-coherence [4] and relay [20]. In order to compare with their results, a similar test was ran with the Yam method. An investigation of the area $0.1 \leq \theta/\tau \leq 3.0$, $0.1 \leq S/(2J) \leq 7$ gave ρ_3 values all grater than $0.9 \gg 0.25$. (Stiction was present in all the cases). These results are encouraging, because no uncertainty region is observed (would imply that $\rho_3 \to 0.25$). Note that we here use MV rather than PV, so the Yamashita method has more information available than the methods tested in [20].

Table E.2: Results altering the filter constant τ_F and the sampling time T_s. The upper table shows simulations with a sticky valve, whereas the lower table shows simulations without stiction. The numbers are the values of ρ_3 (ρ_1).

		τ_F/τ, stiction present	
		0.1	1
T_s/τ	0.1	0.10 (0.40)	0.13 (0.68)
	1	0.08 (0.39)	0.46 (0.77)
	10	0.19 (0.31)	0.37 (0.43)

		τ_F/τ, stiction not present	
		0.1	1
T_s/τ	0.1	0.10 (0.40)	0.12 (0.40)
	1	0.08 (0.39)	0.08 (0.39)
	10	0.03 (0.53)	0.13 (0.47)

In this section the sampling time was set to 0.2 time-units, which may be too low for practical purposes. The next section includes a discussion on the effects of altering the sampling time.

E.4.2 Adding noise

Noise was added to the measurements of MV and OP to investigate the performance with noise present. The method should be sensitive to noise as we use the derivative for finding the symbolic representations. In this section the amount of noise that makes the method inefficient was attempted to be identified.

The "band-limited white noise"-block in Simulink was used to simulate the presence of measurement noise. The noise was filtered with a first order filter $1/(\tau_F s+1)$. The noise power was tuned to be about equal for the OP and MV measurements. This resulted in a noise to signal ratio of about 0.1 to 0.2 for both signals. Then simulations were conducted altering the filter time constant τ_F and the sample time T_s, keeping the noise power block unchanged. Results of running the Yam method on the simulated data are shown in table E.2. To investigate the robustness of the method the indices were also calculated for loops without the presence of stiction.

The most important observations are:

- The frequency-content of the noise is significant. Adding much high-frequency noise makes the method unable to detect stiction. This is evident by looking at the column in table E.2 with $\tau_F/\tau = 0.1$. The results for simulations with and without stiction are about equal.

- The sampling time is important. Lowering the sampling time makes the method inefficient, as the calculation of the differentials will be too dominated by the noise. Setting the sampling time very high is also disadvantageous, so there must be an optimum where we avoid sampling too much noise but still observe the stiction induced limit cycle. From these data, setting the sample time equal to the dominant time constant seems to be a good

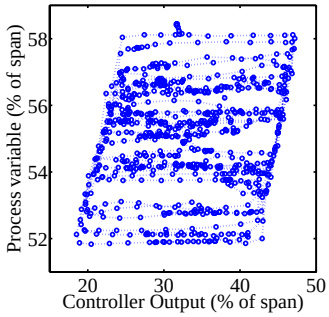

Figure E.5: Loop affected by both stiction and set point changes. From plant data. 720 samples are plotted.

default setting. Theoretically the sampling time should be in the frequency domain of the limit cycle and not of the noise.

- For the cases studied with no stiction in the loop, ρ_1 is always too high, whereas ρ_3 correctly rejects stiction in all cases. This implies that we should use ρ_3 as the determining index, not ρ_1.

From these observations one can conclude that the method is sensitive to noise, and there exists an optimal sampling time.

E.4.3 Varying set-points

An inner PID controller in a conventional cascade is an example of a controller where the set point can be subject to frequent changes. A loop with severe stiction and subject to rapid set point changes found in plant data is shown in figure E.5. Another typical example is a PID controller receiving commands from an advanced process control system (APC). A reasonable time-scale separation between the control layers in a hierarchical structure should be about 5 or more in terms of closed loop response time [26]. When the frequency of set point changes increases from low frequencies to higher, the indices are expected to decrease, because the controller needs to work more to follow the command signal, and hence the stiction pattern will be less clear.

Fig. E.6 shows the calculated indices while varying the frequency of set point change for a loop with parameters $\{S, J, k, \tau, \theta\} = \{6, 1, 1, 10, 10\}$. A SIMC-tuned controller with an assumed closed-loop bandwidth of $\omega_B \approx 0.5/\theta$ [26] was used. For a well designed cascade we expect set point changes at a frequency lower than $(1/10)(1/\theta)$, which will be the assumed bandwidth of the outer controller. (Proof: Let the inner and outer loops have expected closed loop response times τ_{c1}

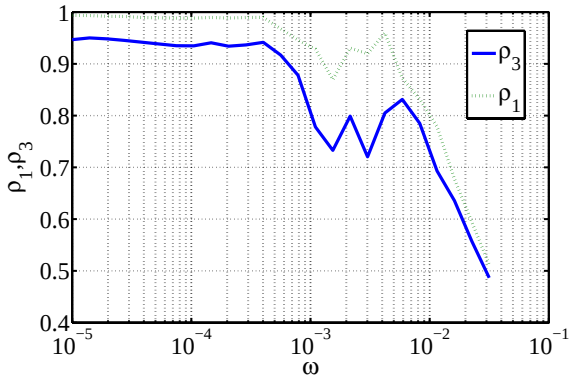

Figure E.6: Indices ρ_1 and ρ_3 as a function of set point-change frequency, $y_{sp} = 0.5\sin(\omega t)$. There was no noise added and $T_s = 0.1\tau$.

and τ_{c2} respectively. Assume that both of them are SIMC-tuned controllers. We then have that the assumed bandwidth for the outer controller is $\omega_B^{\text{outer}} = \frac{1}{2}\frac{1}{\theta_{\text{eff}}^2} = \frac{1}{2}\frac{1}{\tau_{c2}} = \frac{1}{2}\frac{1}{5\tau_{c1}} = \frac{1}{2}\frac{1}{\theta_{\text{eff}}^1} = \frac{1}{10}\frac{1}{\theta}$. θ_{eff}^1 and θ_{eff}^2 are the effective delays in the inner and outer loops.) So, in the case study we expect set point-changes with maximum frequency of about $(1/10)(1/\theta) = (1/10)(1/10) = 0.01$ radians/second. By looking at the fig. E.6 one observes that the indices are relatively high up to this expected frequency, where a decrease in the indices follow for higher frequencies. The stiction parameters were the same in all the cases, but the presence of high-frequency set point changes makes the indices decrease.

This analysis shows that for well-tuned cascades, linearly changing set points within the bandwidth should not be able to deteriorate the performance of the Yam method significantly. If the loop is affected by higher frequency set point changes than what it was designed for (or sustained changes around the bandwidth frequency) the Yam method may not be able to detect a possible presence of stiction.

An advanced process control system (APC) can give commands to the layer below in a step-wise fashion. For fast loops, such as flow loops, these steps may propagate to steps in the OP and MV. A simulation was conducted to investigate this effect. If was evident that introducing steps in the input made the steps dominate the differentials of the recored signals. In words, the signals were only increasing (I) or decreasing (D) when the steps occurred. To understand the significance of this observation to industrial usage, investigation of plant data are necessary. The sharp steps introduce problems, but if they occur rarely or filtered it may still be possible to use the method.

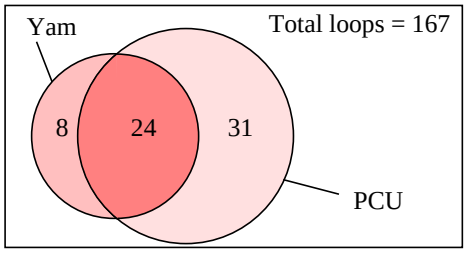

Figure E.7: Loops found to be sticky by the Yam method and the PCU.

Table E.3: PCU report for the 8 loops where Yam reported stiction (see fig. E.7).

Verdict by PCU	Number of loops
Good performance	1
No dominant frequency	7

E.4.4 First conclusions about the technique

For the noise-free case the technique performs well. As noise is introduced, performance decreases. The sampling time may affect the performance of the method. Setting the sampling time too low (or too high) introduces problems for stiction detection. For cascaded loops, simulations shows that as long there is a time-scale separation of 5 or more between the layers, the method should still be able to detection the presence of stiction in the inner loop.

The real test of the method will be plant data, then one will know if the noise level is too high for application of the method or not.

E.5 APPLICATION TO PLANT DATA

A total set of 216 industrial PID loops, of which 167 were flow loops, were available for analysis.

All the loops analyzed were compared with a program called Plant Check-Up (PCU), a prototype for stiction detection in industrial use. The architecture of PCU is shown in [22]. For stiction detection it uses the cross-correlation method [9], the bi-coherence method [4] and the relay technique [20].

E.5.1 Results

In the industrial data set of 167 flow loops the Yam method reported stiction in 32 of the loops, while running the PCU on the same data resulted in 55 loops reported

as sticky. This is illustrated in fig. E.7, where one also observes that 8 loops were found to be sticky by the Yam method but not by the PCU.

Table E.3 shows the report from PCU these 8 loops. When the PCU reports "No dominant frequency" it can not find a dominant frequency of the signals and it does not initiate the stiction detection module. The relay technique requires this frequency. By running only the bi-coherence method, which does not require a dominant frequency, all of these 7 loops were reported to be sticky, therefore these loops can be considered sticky.

For the loop reported to be performing good, data for more weeks were available. For other weeks, the loop was reported to be under presence of stiction by the PCU, therefore this loop is a limit case.

Of the remaining $168 - 33 = 135$ loops for which the Yam method did not report stiction, 31 were found to be sticky by the PCU. If the PCU is regarded as being correct, one can say that the Yam method detects stiction in about half of the cases where PCU detects stiction. The PCU is more advanced, as it has 3 methods implemented for stiction analysis, so it is expected that not all cases can be detected by the simple Yam method.

A visual inspection of the data can be performed on a computer by displaying the recored data in a (OP, MV) plot that evolves with time. Using this tool, it was evident that for the cases where the Yam method reported stiction the expected pattern as shown in fig. E.1 was shown.

Several phenomena were observed for the 31 loops where only PCU found stiction (see fig. E.7). For some of the loops the signals were distorted by noise and no clear patterns were observed. For other loops clear patterns could be observed, but their properties were not of the typical stiction pattern type (see fig. E.1). Often the patterns were similar to an ellipse with no clear parts where the OP was increasing or decreasing with steady MV. A physical explanation of a pattern found in two of the loops is given in section E.5.5, last point.

E.5.2 Sampling time

Application of the method on simulated data showed that there were both lower and upper limits on sample time. The lower is due to sensitivity to noise. For all the loops the sample time was originally 10 seconds. Let x be a vector of observations with sample time T_s. A naive way to simulate a sample time of $2 \cdot T_s$ is to use every second point in x as a basis for calculation of the indices.

Table E.4 shows the result of increasing the sample time by a factor 3 and 6. One observes that the method is a bit sensitive to alterations of the sample time, considering that the number of loops detected changes. However, by inspection of the loops reported to be sticky after altering the sample time, both visually and

Table E.4: Effects of altering the sampling time.

$T_s/T_s^{\text{original}}$	Loops with $\rho_3 > 0,25$
1	32
3	38
6	34

Table E.5: Loops reported to be sticky while changing the length of the observation window

Maximum number of samples	$\rho_3 > 0.25$
total length available	32
2000	30
1000	28
500	29
100	30

with the PCU, the conclusion was the same for the sample times $T_s/T_s^{\text{original}} \in \{1,3,6\}$. When the Yamashita method reports stiction, a visual inspection of the (OP, MV) plot shows clear signs of stiction. The loops that were changed from not being sticky to sticky by increasing the sample time, due to their increase in ρ_3 to above 0.25 from below, had the same stiction-pattern properties as the original 33 loops.

E.5.3 Minimum observation window

It is interesting to quantify the sufficient length of the observation window, as this is an important practical issue when using the method in applications. The typical length of observation for the plant data was about 6000 samples with an interval of 10 seconds, corresponding to an observation window of 17 hours.

From table E.5 one sees that there is not much differences in the results by lowering the allowed samples from unrestricted down to 500 or 100. With a sample time of 10 seconds we could consider sampling 720 samples, corresponding to an observation window of 2 hours.

E.5.4 Noise level in the data

By eye-inspection the noise level in the data seemed to be changing form loop to loop and sometimes also from day to day. For the loops where the method detected stiction the noise level was so low that a clear stiction pattern was evident. For most of the cases where the Yam method failed to detect stiction, it was evident

that the stiction pattern was distorted by a high noise level.

A simple quantification of noise is to divide the apparent amplitude of the noise by the amplitude of the underlaying signal. Doing this, for the loops where the Yam method detects stiction, a typical noise level was about 0.1. This level of noise typically a bit smaller than the noise-level used for simulation. (See table E.2).

E.5.5 Other phenomena observed in the plant data

- A lot of the loops for which stiction was detected had varying set points. The frequency observed for set point changes was typically 0.02 radians/second or lower. Fig. 2.6 shows how the sensitivity for detection varies with set point changes. It is difficult to draw direct parallels to this figure from the plant data under investigation, but it seems as if the cascaded loops in the current data had a frequency of set point-changes low enough to enable usage of the Yam method.

  For PID loops receiving commands from APC there is still work to do. For the time of writing it is uncertain if the presence of steps for the APC affects the method significantly or not. Until further research is conducted it is recommended to avoid analysis of data were the set point changes non-linearly with a high frequency.

- When applying the method on industrial data it was observed that saturation of the valve can be wrongly regarded as stiction by the method. To avoid this some saturation detection in the implementation should be included. With the simple implementation presented in [34] there is not need for the ranges of controller or flowrate. This must be included in the saturation-handling.

- For two of the loops where stiction was not detected by the Yam method another kind of stiction pattern was observed. Here the maximum change in controller output was not when the valve was stuck, but when it was jumping after being stuck. Consider the time-derivative of the output of a typical PI controller: $\partial u(t)/\partial t = K_c\left(\partial e(t)/\partial t + \tau_I^{-1}e(t)\right)$, where K_c is the controller gain, τ_I is the integral time, and $e(t)$ is the control error. For the two loops, $|\partial e(t)/\partial t|$ *when the valve is jumping* was larger than $\tau_I^{-1}|e(t)|$ *when the valve is stuck*. In the symbolic representation of the OP this can lead to OP being increasing or decreasing *when the valve is jumping* and steady *when the valve is stuck*. Hence the method will not work in this case.

E.6 CONCLUSIONS

The objective of the paper was to investigate the reliability of the recently proposed Yam technique [34] for automatic recognition of stiction patterns in oscillations recorded in industrial data.

A first limitation is that the technique is based on values of the controlled variable (OP) and manipulated variable (MV), which are available only in the case of intelligent valves or for the special case of flow loops; anyway from the analyzed set of more than 200 industrial data sets, these last constitute a relevant number, amounting to about $(167/216 \approx)\ 3/4$ of actual control loops.

From the investigation of several points of interest in the light of industrial applications, the following conclusions can be drawn:

- The method is based on derivatives and uses the standard deviation as threshold for the symbolic identification of stiction pattern: some sensitivity to the level of noise is shown in simulation. The low level of noise encountered in most of industrial registrations makes this drawback less relevant.

- The observation window can be limited to few periods of oscillations and, for sampling time $T_s = 10$ seconds, a default of 2 hours can be suggested.

From the comparison of results on industrial data with a package which performs a sequential application of stiction detection methods, it can be concluded that stiction is recognized in about 50% of cases. Therefore the Yam technique can be suggested for a fast identification of sticky loops with clear patterns, leaving more difficult loops for a deeper analysis, with two advantages: a quick detection of the onset of stiction and a save of computation time. This characteristics, together with the ease of implementation of the algorithm with any simple language, makes the technique suitable for on line implementation.